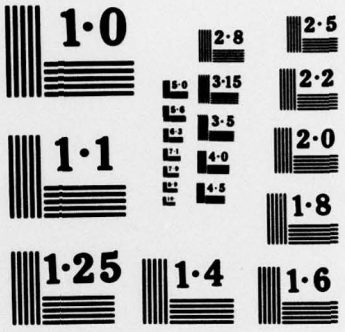
AD-A040 806 MODELS OF THE US ARMY WORLDWIDE LOGISTIC SYSTEM (MAWLOGS) VOLUME III MODU..(U) BDM CORP VIENNA VA FEB 77 BDM/W-76-211-TR-VOL-3-PT-8-9 DAAG39-76-C-0134 1/2

UNCLASSIFIED F/G 15/5 NL


1·0
1·1
1·25
1·4
1·6
2·8
3·15
3·5
4·0
4·5
2·5
2·2
2·0
1·8

AD A040806

DDC
RECEIVED
JUN 22 1977
A

MODELS OF THE US ARMY WORLDWIDE LOGISTIC SYSTEM (MAWLOGS)

VOLUME III - MODULE CATALOG

PART 8 - CONTAINERIZATION MODULES

PART 9 - MODEL CHANGE MODULES

AD No.
DDC FILE COPY

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

(1)

THE BDM CORPORATION

1920 Aline Avenue
Vienna, Virginia 22180
Phone (703) 821-5000

(12) 157 p.

(11) FEB[redacted] 1977

(14) BDM/W-76-211-TR-Vol-3-Pt-8-9

(15) DAAG 39-76-C-0134

DDC
RECEIVED
JUN 22 1977
A

(6)

MODELS OF THE US ARMY WORLDWIDE LOGISTIC SYSTEM (MAWLOGS)

VOLUME III - MODULE CATALOG

PART 8 - CONTAINERIZATION MODULES

PART 9 - MODEL CHANGE MODULES

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

1.

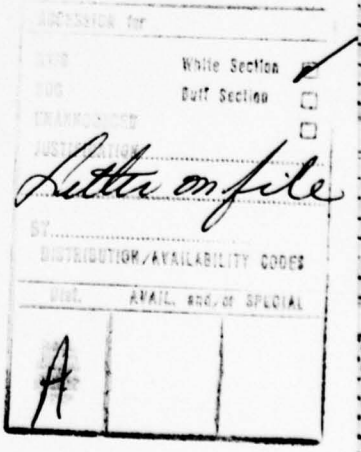
391962

FOREWORD

This Part of the MAWLOGS Module Catalog is submitted to the Department of the Army, Washington, D.C. 20310 by the BDM Corporation, 1920 Aline Avenue, Vienna, Virginia 22180, as required by Contract Number DAAG39-76-C-0134.

This document is one of sixteen that describes the Models of the US Army Worldwide Logistic System (MAWLOGS). MAWLOGS was developed for the Deputy Chief of Staff for Logistics, Department of the Army, under the monitorship of the US Army Logistics Evaluation Agency and the US Army Logistics Center. The development objective was to provide a capability to analyze and compare the performance of multifunctional logistic systems, to include both current and proposed systems. MAWLOGS is not a model of a particular Army logistic system. It is a system for the rapid assembly of discrete-event stochastic simulation models of a wide range of logistic systems and for the processing and interpretation of data associated with the execution of such models. The original documentation was completed in 1974. Documentation for subsequent software development has added five volumes to the original eleven. The documents describing the system and how to apply it are listed below.

Volume I - General Description
Volume II - User's Manual
Volume IIA - Addendum to User's Manual
Volume III - Module Catalog
- Part 1 - Service Modules
- Part 2 - Field Maintenance and Supply Modules
- Part 3 - Wholesale Supply and Maintenance Modules
- Part 4 - Transportation Modules
- Part 5 - Communication Modules
- Part 6 - Continuous Service Modules
- Part 7 - Continuous Supply Modules
- Part 8 - Containerization Modules
- Part 9 - Model Change Modules



Volume IV - Programmer's Guide

Part 1 - Writing and Testing Modules, Module Library Maintenance and General Guidance

Part 2 - Technical Description of the Model Assembler Program

Part 3 - Technical Description of the Output Data Postprocessor System and Programs

Part 4 - Module Library Program Listings

Volume IVA - Addendum to Programmer's Guide

TABLE OF CONTENTS

Chapter		Page
	FOREWORD	iii
	TABLE OF CONTENTS	v
	LIST OF FIGURES	ix
	LIST OF TABLES	xi
	PART 8 - CONTAINERIZATION MODULES	
I	INTRODUCTION TO MODULE CATALOG	I-1 to I-9
	A. Catalog Layout	I-2
	1. Module Descriptions	I-2
	2. Module Families	I-2
	3. Data Structure	I-3
	B. Catalog Conventions	I-3
	1. Upper Righthand Corner	I-4
	2. General Description	I-5
	3. Assembler Inputs	I-5
	4. Examples	I-6
	5. Statistics Collected	I-6
	6. GASP Files Used	I-6
	7. Permanent Attributes Accessed	I-7
	8. Verb Inputs	I-8
	9. Verb Outputs	I-9
	10. Programs Called	I-9
	11. Input/Output Files Used	I-9
II	CONTAINERIZATION MODULE FAMILY DESCRIPTION	II-1 to II-11
	A. Overview	II-1
	B. Verb Use	II-4
	1. CCP With Discrete Transportation	II-4
	2. CCP Without Discrete Transportation	II-8
	3. Full Container Assignment at Depots	II-8
III	CONTAINERIZATION DATA STRUCTURE	III-1 to III-17
	A. Dataset Descriptions	III-1
	B. Shipment Representation	III-1
	C. Statistics Codes	III-14

TABLE OF CONTENTS (CONTINUED)

Chapter		Page
IV	CONTAINERIZATION VERBS	IV-1 to IV-58
ADLVR	IDENTIFY AN ACCOUNT FOR CONTAINER DELIVERY	IV-3
CASIN	ASSIGN A CONTAINER LOAD FOR A CLUSTER	IV-5
CCP	CONTAINERIZATION AND CONSOLIDATION POINT	IV-9
CDLVL	CONTAINER DELIVERY-EMPTIES CONTAINERS INTO A CDLEVL DATASET	IV-13
CDP	CONTAINER DELIVERY INPUT	IV-15
CLOAD	CONTAINER LOADING OF ASSIGNED SHIPMENTS	IV-17
CLOSS	CONTAINER LOSSES	IV-21
CONRP	REPORT ON CONTAINER OPERATIONS	IV-25
DASIN	ASSIGNA CONTAINER LOAD FOR A DROP POINT	IV-33
FCASN	FULL CONTAINER ASSIGNMENT AT DEPOTS	IV-37
RCVCD	RECEIVE CONTAINERS AT A CONTAINER DELIVERY POINT	IV-41
RCVCP	RECEIVE FLOW OF MATERIAL AT A CONTAINERIZATION POINT AND CONVERT TO DISCRETE SHIPMENTS	IV-45
ROUCS	CALCULATE ROUTING FOR A CONTAINER LOAD OF SHIPMENTS	IV-49
SNDCP	CONVERT DISCRETE SHIPMENTS TO FLOW AND SEND FROM A CONTAINERIZATION POINT	IV-53 IV-57

PART 9-MODEL CHANGE MODULES

V	MODEL CHANGE MODULES FAMILY DESCRIPTION	V-1 to V-9
	A. Overview	V-1
	B. Data Change Modules	V-1
	C. Staging	V-3
	D. DTABLE DATA HANDLING	V-8
VI	MODEL CHANGE VERBS	VI-1 to VI-36
CANCL	VERB TO CANCEL A GASP FILE ENTRY	VI-3
CHPAR	INPUT VERB TO CHANGE DISTRIBUTIONS IN PARAM ARRAY DURING A MODEL RUN	VI-5
CHPDS	INPUT VERB TO CHANGE PDS DATASETS DURING A MODEL RUN RUN	VI-11
CHSTG	CHANGE STAGING DURING A RUN	VI-19
INSTG	PRINT CONTENTS OF PUSHDOWN STACKS	VI-23
PRSTK	PRINT CONTENTS OF PUSHDOWN STACKS	VI-25
RFLON	SETS UP A RETURN FLOW OF A DISCRETE SHIPMENT WITHOUT RELEASING THE RETS POINTER OR THE STACK ENTRY	VI-27

TABLE OF CONTENTS (CONTINUED)

Chapter		Page
RPSTG	MODULE TO REPORT STAGING DATA	VI-31
STGBR	BRANCH TO PROPER PS FOR CURRENT STAGE	VI-33
WBFIL	COMPUTES WEIBULL FUNCTION VALUES FOR A GIVEN VALUE	VI-35
VII	MODEL CHANGE ROUTINES	VII-1 to VII-30
CANCEL	CANCEL A GASP FILE ENTRY	VII-3
CPLAEN	PLACE AN ENTRY IN AN /IZHOLD/STACK FOR FIFO OPERATION	VII-5
DELDS	DELETE A SERIES OF PDS DATASETS WITH COMMON COORDINATES	VII-7
DTBGC	ARRAY DTABLE GARBAGE COLLECTION IN COMMON/TBLCOM/	VII-9
DTPNEW	NEW DTABLE POINTERS AFTER GARBAGE COLLECTION	VII-11
INDTB	INPUT ENTRIES TO DTABLE ARRAY	VII-13
IXZNOD	RETURNS INDEX OF NODE WITH GIVEN NAME IN ARRAY ANODEL	VII-15
MLTL	MULTIPLY THE VALUE OF AN ELEMENT IN A PDS DATASET BY A FACTOR	VII-17
MSTAGE	ADVANCES STAGE NUMBERS FOR SEQUENCES WHOSE NEXT STAGE TIME IS TNOW AND SCHEDULES NEXT STAGE EVENT	VII-19
RDNDS	READ A GROUP OF PDS DATASETS WITH THE SAME VALUES	VII-21
RLDTB	RELEASES SPECIFIED ENTRIES FROM DTABLE	VII-23
RSETPA	RESETS POINTERS IN ARRAY PARAM TO ARRAY DTABLE AFTER GARBAGE COLLECTION	VII-25
RSETRD	RESETS RATE/LEVEL DELAY BOXCAR POINTERS AFTER DTABLE GARBAGE COLLECTION	VII-27
RSETRS	RESETS DTABLE POINTERS FROM LDSC, EXRAT, AND DTABLE GARBAGE COLLECTION	VII-29

THIS PAGE LEFT INTENTIONALLY BLANK

LIST OF FIGURES

Figure	Title	Page
II-1	Rate/Level and Discrete Interface	II-2
II-2	CCP Node with Discrete Transportation	II-6
II-3	Container Delivery Point Node	II-7
II-4	CCP Node Without Discrete Transportation	II-9
II-5	Full Container Assignment at Depot Examples	II-10
V-1	Example Node ZINIT With Data Change Capability	V-4
V-2	Example of Node With Staging	V-5

THIS PAGE LEFT INTENTIONALLY BLANK

LIST OF TABLES

Table	Title	Page
II-1	Containerization Verbs	II-5
III-1	Containerization Datasets	III-2
III-2	Labeled COMMON /CONCOM/	III-3
III-3	ACCLOC Dataset	III-4
III-4	AGGACC Dataset	III-4
III-5	AGGSF Dataset	III-5
III-6	CCPPOL Dataset	III-5
III-7	CCPSTS Dataset	III-6
III-8	CDLEVL Dataset	III-7
III-9	CDSTAT Dataset	III-7
III-10	CLOSS Dataset	III-8
III-11	CLUSTR Dataset	III-9
III-12	CONPOL Dataset	III-10
III-13	CONSTS Dataset	III-10
III-14	CONTNR Dataset	III-11
III-15	CONTYP Dataset	III-11
III-16	DPOFAC Dataset	III-12
III-17	DPOLVL Dataset	III-12
III-18	DROPPT Dataset	III-13
III-19	PRIORT Dataset	III-13
III-20	Container Attribute Set	III-15
III-21	Containerization Statistics	III-16

LIST OF TABLES (CONTINUED)

Table	Title	Page
III-22	Statistics Type Codes (fedcba)	III-17
V-1	Model Change Modules	V-2
V-2	Common BLOCK /ZSTAGE/	V-7
V-3	Common BLOCK /DTBMAP/	V-9

THE BDM CORPORATION

VOLUME III - MAWLOGS MODULE CATALOG

PART 8 - CONTAINERIZATION MODULES

CHAPTER I

INTRODUCTION TO MODULE CATALOG

The MAWLOGS System is a set of computer programs by which a large number of stochastic, discrete-event, computerized simulation models of logistic systems or portions of systems can be generated. The computer programs in the MAWLOGS System may be divided into four groups, each of which constitutes a basic element of the System. One, called the Module Library, contains programs that simulate logistic activities or perform simulation bookkeeping activities. A second group of programs, called the Model Assembler, generates MAWLOGS models by retrieving required modules from the Module Library and creating sufficient additional programs to link the selected modules together into a model. The other two groups constitute the Output Data Postprocessor System and the Automated Input Data System. An overview of the MAWLOGS System is given in Volume I and, in briefer form in Section I of Volume II.

The MAWLOGS Module Catalog is the directory to the Module Library. Its purpose is to describe to the user of the MAWLOGS System the modules that are in the library and the logistic or simulation function each performs. For logistic modules, such additional information as the level of detail, decision rules used, and statistics collected are given.

The User's Manual, Volume II of the four-volume MAWLOGS documentation, describes how to define a model to the Model Assembler in terms of modules, how to run the Assembler to produce the model, how to use the model, to include preparation of inputs for simulation control, and how to use the MAWLOGS Output Data Postprocessor to analyze model outputs. Volume IV, "Programmer's Guide," contains technical descriptions of the Model Assembler and the Output Data Postprocessor System. The Programmer's Guide also discussed the writing of modules.

Thus, the user of the Module Catalog is likely to be working from either of two directions. The more usual case will be that of the model writer using the User's Manuals as a guide and the Module Catalog as a

reference from which to select modules appropriate for use in his model. The other case is that of the module writer who wishes to write new modules or modify existing ones. The module writer will be using the Programmer's Guide as his guide, and the technical details in the Module Catalog to determine what to modify in existing modules or what specific points must be accommodated to make any new modules compatible with existing modules.

A. CATALOG LAYOUT

The MAWLOGS Module Catalog is organized into Parts, each of which, with the exception of Part 1, describes the modules for a logistic functional area. Part 1 describes the simulation service modules. In the initial catalog, Part 2 covers field maintenance and supply, Part 3 covers wholesale maintenance and supply, Part 4 covers transporation, and Part 5 covers communications. A Part is organized into four major sections in addition to this introduction. Sections 4 and 5 describe the modules. Sections 2 and 3 present information common to large groups of modules under the headings MODULE FAMILIES and DATA STRUCTURE.

1. Module Descriptions

It is essential to distinguish between two types of modules -- those that may be referenced in a model description, called verbs, and those that may not, called routines. The catalog descriptions of verbs for a logistic functional area will be found in the fourth section of a Part, entitled "VERBS," the descriptions of supporting routines in the fifth, entitled "ROUTINES." Within each section the modules are in alphabetical order. Each Part contains an index to its verbs and supporting routines.

A module of either type is documented in a standard format, using a certain amount of phraseological and notational convention. The format and conventions are described further in subsequent paragraphs.

2. Module Families

The presence of a section called "MODULE FAMILIES" reflects recognition of the fact that modules are designed in groups along certain lines

chosen by the designer. Thus, the designer chooses the scope and level of detail with which he will cover a set of logistic activities, to what degree he will "modularize" the coverage, which options he will provide, the data structures he will use, and a variety of other details in the programming approach. He will base his work on one or more major assumptions and objectives. A knowledge of the background and viewpoint underlying a group of related modules can greatly facilitate their proper use. The MODULE FAMILIES section is intended to provide this background and overview. Examples of nodes or subnodes in which the verbs of a family are used in their intended patterns should also be sought in the MODULE FAMILIES sections. The types of statistics collected in a module family are tabulated all together in this section.

3. Data Structure

The section called "DATA STRUCTURES" contains descriptions of the organization of the data storage areas that support a module family. The description consists of the names of common blocks and, for each, the names and definitions of variables. Highlights and key points of the data structure are described. Normally there will be a major section to describe permanent attributes and a lesser one to describe temporary attributes.

B. CATALOG CONVENTIONS

A verb description in the catalog follows the outline shown below.

		VERBNAME
	simple nonsimple	/function/family
VERBNAME (arguments)		month/year written
General Description		
Assembler Inputs		
Examples		
Statistics Collected		
GASP Files Used		
Permanent Attributes Accessed		
Verb Inputs		
Verb Outputs		
Programs Called		
Input/Output Files Used		

Supporting routines are described in a similar format with the inapplicable headings "Parameter Slots" and "Examples" omitted. Note that a listing of the computer program is not included; these are available in Part 4 of Volume IV, Programmer's Guide. Further discussion of the content of each section of the outline, to include particular notational or descriptive conventions used, are given next. The sections progress from general to detailed. For most verbs, the model writer should not have to go beyond the section called "Statistics Collected."

1. Upper Righthand Corner

Here a coded summary of the verb is given in two lines under its name. The first line identifies the verb as simple, S, or nonsimple, N. It also gives a code for the logistic or other function represented and, following this, a code for the module family it is considered to be a part of. Values for the function and family codes are given in the accompanying tabulation. The second line under the verb name gives the month and year when written.

Functional areas		Module families	
Code	Area	Code	Family
C	communications	C1	communications
LS	logistic service	CH	change logic
M	maintenance	CS	continuous service
R	rebuild	LS1	logistic service
S	supply	M1	field maintenance
SS	simulation service	M2	wholesale maintenance
ST	statistics collection	PDS	permanent datasets
T	transporation	R1	field rebuild
U	user	S1	field supply
V	salvage	S2	wholesale supply
X	direct exchange	S3	revised field supply
		S4	continuous supply
		SS1	simulation service
		ST1	statistics collection
		T1	aggregate transportation
		T2	detailed transportation
		T3	containerization
		U1	fleet user
		V1	salvage
		X1	field direct exchange
		X2	revised DX

2. General Description

This section contains a general discussion of what the verb does, to include any significant mathematics. The flow of control when execution of the verb is completed is also described. The phrases "to the calling program" and "to the time file" are used for the flow of control when, respectively, a CALL RETLOG returns control to the previous stage of the logic flow that led to the verb, or a CALL ZFADE interrupts the present flow and returns control to program MAWGSP, which then removes the next event from the time file. For modules which read cards, write reports, or read or write other external files, the card formats, report formats, or file formats are described.

3. Assembler Inputs

Here the information to be specified for the verb when it is referenced in a model description written for input to the Model Assembler is defined. Two types of information are identified, arguments and parameter slots. Arguments are variables for which values are specified in the form "P = i, j, ..." i.e., they are variables whose names will appear as arguments of the verb in its FORTRAN subroutine form. Parameter slots, abbreviated "PS," are places in the logic of a verb where access to node references and the logic of other verbs may be provided for by the Model Assembler. The content of a parameter slot is described in general terms. It will be recalled from the User's Manual that a PS need not be filled. If it is filled, it is incumbent on the user to insure that the verbs chosen as content of the PS are compatible with each other and with the verb in whose PS they appear. In particular, this means that the verbs chosen for the PS must not only represent the general functional content required, but that their inputs as described in the section of this outline called Verb Inputs must be compatible with the outputs of the using verb described in the section of this outline called Verb Outputs and vice versa.

Some parameter slots are for sending something out -- a message or a shipment, for example. Such a parameter slot will often be marked "D-delay" or "R-delay." These designations represent cases in which, respectively, (a) control is expected to be given to the time file after

the arrival event has been scheduled, and (b) control is to be returned to the verb containing the PS after the arrival event has been scheduled. The prototypical verbs of these types are DELAY and RTURN. The corresponding communications verbs are COMMD and COMMR, and the corresponding transportation verbs are TRAND and TRANR. An important point to note is that in the case of a PS marked "R-delay, " the presence of a RTURN or similar verb is mandatory, even if the delay time is to be zero, while in a "D-delay" PS, the DELAY or similar verb may be omitted if the delay is to be zero or if some other disposition of the entity normally thought of as being "sent" is to be provided for.

4. Examples

Examples of how the verb is used in portions of model descriptions are shown or described. Certain limitations or options are often highlighted under this heading.

5. Statistics Collected

The variables on which statistics are collected are listed in terms of a descriptive phrase and, for variables observed through calls to STATI, the names of the variables and common blocks in which the statistics indicator codes and indexes are stored. However, the statistics type code, resource identifier, and node code by which a statistics variable is identified are not given here; they must be sought in the Module Family section of the Part of the catalog in which the verb is described.

6. GASP Files Used

The GASP files accessed by the module are identified, primarily in terms of a general description of the content of the file and the names of variables and common areas in which the GASP file numbers are stored. The indirect identification of the file number reflects the fact that in general the file numbers are assigned to a particular queue when the first entry is to be filed or when the logistic functional area is initialized. Thus, the file number assignments will vary among models and runs of the same model. When a file number is stored in PERMAT in an attribute set of type QUEUE(fcntyp), the form "QUEUE(fcntyp).n" is used to identify the location where a file number is stored. In this notation n is the index

of the element in the attribute set that contains the GASP file number and "fcntyp" is a function type code such as MNOD or SNOD. The attributes of a file entry and the ranking scheme for a file are defined under the module description headings Verb Inputs and Verb Outputs.

7. Permanent Attributes Accessed

This and the next two headings -- Verb Inputs and Verb Outputs -- are related in that they all deal with data transmission among modules and between modules and the data storage areas. Thus, an initial point is that they all deal with internal data transmission as distinguished from card inputs or other input or output involving external files.

In discussing this internal data flow, it is useful to distinguish between permanent and temporary data. The phrase "permanent data" refers to data stored in locations whose variable names and the system characteristics they represent are fixed throughout a run. All of the data in PERMAT, in other permanent attribute storage areas, and in the statistics counters are of this type. The phrases "temporary data" refers to all data that are not permanent. It includes such things as the content of array POOL through which PERMAT attribute sets are transmitted, the content of arrays ATRIB and HOLD through which attributes of temporary entities stored in the GASP filing array are transmitted, and the variables IQ, KRA, and KRB, used to designate GASP file numbers and ranking schemes. Utility variables IZSWT and ZSWT in /ZMAWSY/ are also often used for temporary data. The dynamic storage areas in which the GASP files and MAWLOGS stacks are stored also contain temporary data, but these areas are accessed directly only by a few simulation service routines and so are not often referred to in the module descriptions.

The term "permanent attributes" refers to those elements of permanent data that characterize aspects of the logistic system being simulated. Examples are policy parameters such as a reorder point, resource characteristics such as the price of an item, and the capacity and response characteristics of such system elements as transporation links or materiel handling

and storage points. Data such as statistics counters and statistics indicators and indexes, or the GASP file descriptors stored in /GSPFIL/ are examples of permanent data that are not permanent attributes.

Two other terms remain to be defined: "module inputs" and "module outputs." A module input is any variable whose value (a) was determined outside the module and (b) influences the logical path taken in the module or the amount by which a module output variable will be changed. A module output is any variable whose value may be changed by the module. Inputs and outputs may be either permanent or temporary data. Further discussion of inputs and outputs may be found in the next two subsections.

Here, it will suffice to summarize the overall conventions governing the content of the three sections: Permanent Attributes Accessed, Verb Inputs, and Verb Outputs. Permanent Attributes Accessed lists all permanent attributes retrieved by, stored by, or modified by a module. Verb Inputs does not list all inputs; it lists only inputs that are not listed under the Permanent Attributes Accessed and Statistics Collected headings. Verb Outputs lists all outputs, temporary and permanent, including any entries under Permanent Attributes Accessed whose values may be changed by the module. The objective here is to minimize the multiple mention of a variable while retaining the utility of the three headings under discussion.

The purpose of the present heading, Permanent Attributes Accessed, is to list in one place those principle variables, by which the status of the logistic system is represented in the model, that have interplay in a module. From the point of view of the student of the logistic system, it is just this aspect of the module that is of interest. An element of an attribute set in PERMAT is referenced in the following notation -- FUNC(arg) .m, where FUNC is a PERMAT function name representing an attribute set type, such as S!COM, arg is the argument type, such as "item," and m is the index of the element in the attribute set.

8. Verb Inputs

The data described under this heading are a subset of the total internal verb inputs. Categories not described are the statistics indicators and indexes implied by the entries under the heading Statistics Collected

and any inputs under Permanent Attributes Accessed. Inputs that are listed include primarily temporary data. A major type of temporary input is the set of attributes associated with a temporary entity, usually received in arrays ATRIB or HOLD. Inputs are listed under major subheadings indicating their source, such as Calling Program, PS1 (parameter slot 1), Verb DISTR. The purpose of the Verb Inputs section is to facilitate determination of whether the verb will fit in a particular parameter slot and to facilitate the writing of new verbs which may be used in conjunction with this verb.

9. Verb Outputs

The general purpose of this section is similar to that for Verb Inputs -- to facilitate determining whether one verb meshes with another and to facilitate the writing of new verbs that mesh with existing verbs. The outputs are listed under major subheadngs representing their destination -- PS1, PS2, ..., Calling Program, Permanent Attributes, and any other suitable headings such as the names of subroutines or verbs called directly. Only variables the module was designed to affect are listed. Thus, for example, if a set of temporary attributes is input to the module and is not changed by it, they are available for use by a subsequent module and in this sense may be thought of as having been "output" by the current module. But if the current module will never change any of the attributes and is not naturally thought of as "producing" an event of the type represented by the input attributes, these attributes will not be listed under Verb Outputs. The same is true of any other temporary data that are input but not changed.

10. Programs Called

Any programs called directly by the module are named here, with the exception of a few whose use is very common. These are CALLOG, RETLOG, and LINK for stack accessing; STAT1 for statistics collection; and FPOOL, GPOOL, PPOOL, CHPMT, PRMT, and SETPMT for PERMAT accessing.

11. Input/Output Files Used

The files referred to here are external files as distinguished from GASP files. They are identified in terms of their FORTRAN logical file numbers and general contents.

CHAPTER II
CONTAINERIZATION MODULE FAMILY DESCRIPTION

A. OVERVIEW

Containerization modules were designed to simulate the consolidation of shipments into various sized container loads and the unpacking and distribution of the contents of a container to one or more customers. This family of modules may optionally interface with the transportation families in MAWLOGS which simulate the movement of discrete shipments (containers) through a multimode network. The containerization family of modules was disigned to interface with the continuous supply modules and was initially used in the DSS Peace-to-War Transition modules. However, these modules with different interface modules than RCVCP and RCVCD will work with the discrete event supply and maintenance module families. These container loading and unloading points are places in the DSS where the continuous supply simulation and the discrete containerization and transporation simulations interface. Figure II-1 illustrates this interface and how the containerization logic works. A continuous flow of materiel enters the CCP for each DSS account for each class and priority. These flows are translated to discrete shipments and placed in stacks by account and priority, waiting consolidation to a container load. Container loads are formed based on the amount and age of materiel waiting for the customers clustered in a geographical area. Once a load is formed, it may be sent through the transportation network to a container delivery-point (CDP) or immediately converted back to a flow of materiel to the customers. At the CDP, the contents of each container is converted back to flow as a receipt rate for the customer. In either case, the model knows the contents of each container and the utilization of containers in the system. This discrete method of simulating the movement of shipments in containers enables the model to maintain the identity of materiel for each class at each supply support activity.

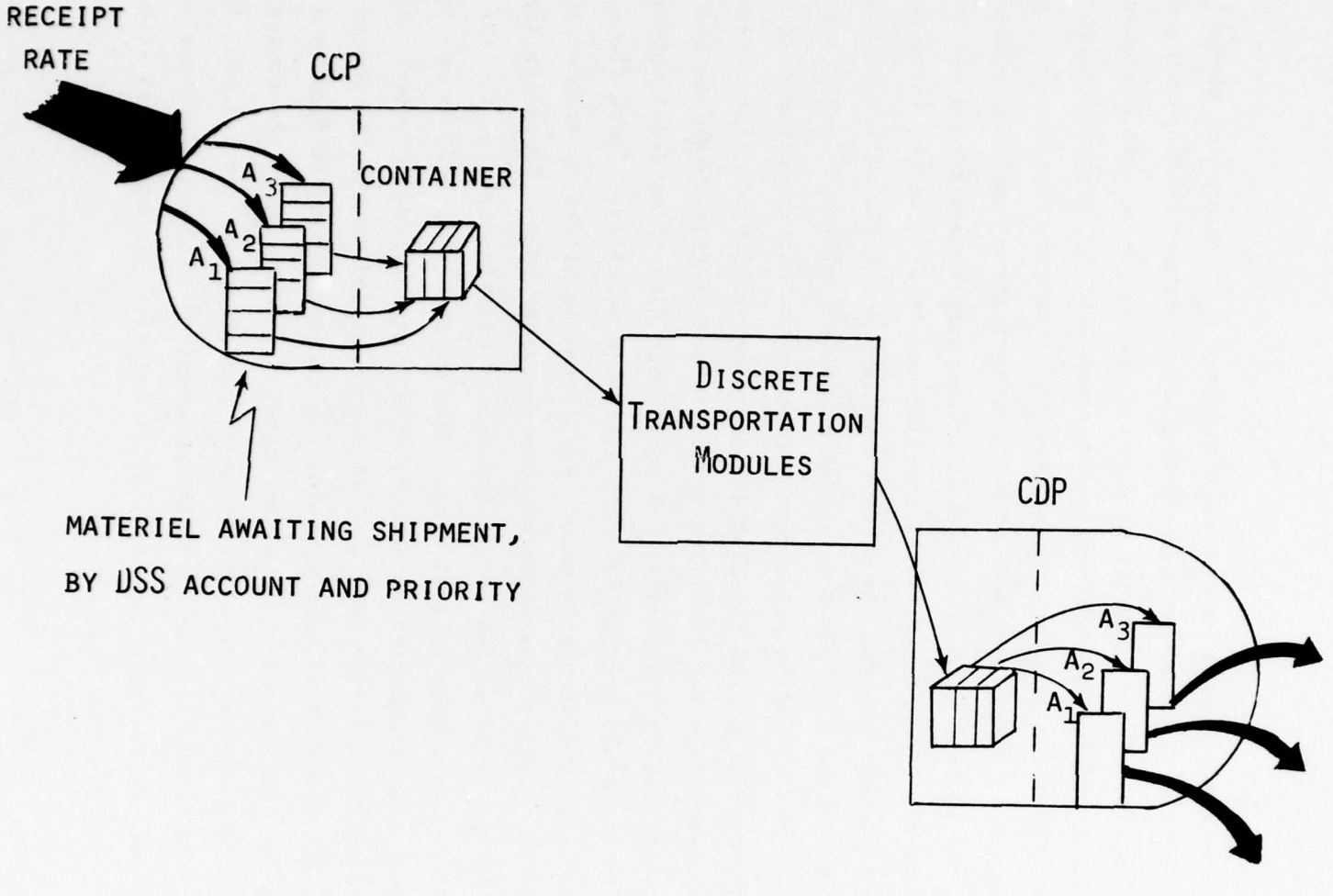


Figure II-1. Rate/Level and Discrete Interface

A list of the activities simulated in the containerization modules and the input parameters which can be varied follows. These two lists indicate the scope and the flexibility of the module family.

(1) ACTIVITIES

(a) Consolidation of Shipments to Accounts/Cluster by Priority

(b) Containerization in Various Size Vans and Pallets

(c) Unloading of Shipments at Accounts or Drop Points

(2) INPUT PARAMETERS

(a) Account Clustering/Drop Points

(b) Container Types

(c) Container Capacities

(d) Container Loading Policies

Maximum Hold Time

Maximum Number of Consignees

Minimum Economic Fill

(e) Priority Scheme.

The clusters to be utilized are specified in the input data and may be altered at any time. The types of containers which can be used are specified and can include any sized surface vans and air pallets. The loading policies for containers at each CCP can be set to indicate the maximum hold time for materiel, the maximum number of consignees in a container, and the minimum fill required for an economic container load. A priority scheme for loading can also be specified to restrict certain priorities to specific container types.

The list below shows what can be measured in the containerization modules. In general, we would like to know what transporation requirements are generated in model runs. In addition, what delays are encountered in moving shipments and what the cause of the delays may be. The measures at the CCP and at each account are listed.

(1) Measures at CCP

(a) Amount of Materiel Waiting for Each Account by Priority

(b) Age of Materiel Waiting

(c) Number of Containers Shipped by Type

(d) Percent Container Utilization by Type

(e) Total Materiel Shipped

(f) Number of Consignees per Container

(g) Number of Uneconomical Containers Sent

(2) Measures of Customers

(a) Number of Containers Received

(b) Travel Time of Containers

(c) Amount of Materiel Received.

The verbs contained in this family are listed in Table II-1. This family includes modules to represent a separate containerization and consolidation point (CCP) as well as full container shipments from a depot. Interface verbs for continuous supply and discrete transportation are also included.

B. VERB USE

1. CCP With Discrete Transportation

The first use of the containerization modules shown is to represent the shipment of containers from a CCP to a demand generator node via a discrete transportation network. Figure II-2 shows a node description for a CCP. Subnode 1 is the interface with a continuous supply flow and is tied into the other continuous nodes (RLCTR is the next one) so that it is executed every time step. The logic in RCVCP translates the incoming flow to discrete shipments for the CCP to consolidate and load. Subnode 2 simulates discrete CCP operations and operates on a separate time cycle from subnode 1. This node assigns shipments to containers based on clusters of accounts through CASIN or drop points in a geographic area through DASIN. Once a container is loaded, it is sent out through parameter slot 4 to the transportation control node TRANS. The transportation control node will deliver the container to a prespecified point. Figure II-3 shows how an account node identifies itself for containerized shipments. and receives those shipments. Only the container related logic is shown in Figure II-3.

TABLE II-1. CONTAINERIZATION VERBS

VERB NAME	DESCRIPTION
ADLVR	IDENTIFY AN ACCOUNT FOR CONTAINER DELIVERY
CASIN	ASSIGN A CONTAINER LOAD FOR A CLUSTER
CCP	CONTAINERIZATION AND CONSOLIDATION POINT
CDLVL	CONTAINER DELIVERY - EMPTIES CONTAINERS INTO A LEVEL DATASET
CDP	CONTAINER DELIVERY POINT
CLOAD	CONTAINER LOADING OF ASSIGNED SHIPMENTS
CLOSS	CONTAINER LOSSES
CONRP	REPORT ON CONTAINER OPERATIONS
DASIN	ASSIGN A CONTAINER LOAD FOR A DROP POINT
FCASN	FULL CONTAINER ASSIGNMENT AT DEPOTS
FCLOD	FULL CONTAINER LOADING AT DEPOTS
RCVCD	RECEIVE CONTAINERS AT A CONTAINER DELIVERY POINT
RCVCP	RECEIVE FLOW OF MATERIEL AT A CONTAINERIZATION POINT AND CONVERT TO DISCRETE SHIPMENTS
ROUCS	CALCULATE ROUTING FOR A CONTAINER LOAD OF SHIPMENTS
SNDCP	CONVERT DISCRETE SHIPMENTS TO FLOW AND SEND FROM A CONTAINERIZATION POINT

```
+
+    CCPE CONTAINERIZATION AND CONSOLIDATION POINT
+
CCPE.     RLNOD (P = *CCPE, 57.),
          RCVCP (P = 1), * RLCTR        + RECEIVE FLOW AT CCP
+
+    SUBNODE 2 - CCP OPERATIONS
+
/2/       CCP       (1 = CASIN     (P = 0)  $     + CONTAINER ASSIGNMENT
                                                    TO ACCOUNTS
                     2 = DASIN     (P = 0)  $     + CONTAINER ASSIGNMENT
                                                    TO DROP POINTS
                     3 = CLOAD              $     + CONTAINER LOADING
                     4 = TRETC     (P = 57  $     + SEND CONTAINER OUT
                                    1 = RTURN (P = 0),
                                        *TRANS      )  $
                     5 = RTURN     (P = 0),       + RESCHEDULE CCP
                         *CCPE .2)  $             + OPERATIONS
```

Figure II-2. CCP Node with Discrete Transporation

```
+
+    ACCOUNT NODEA
+
NODEA.  RLNOD (P = *NODEA, 13.),                    + IDENTIFY NODE
        ADLVR (P = *NODEA, 0  $                     + IDENTIFY CONTAINER
                                                      DELIVERY POINT
               1 = *NODEA .2),
        ACCTA (1 = . . .                            + SUPPLY LOGIC
                          ),
        CDP, *NODE B                                + CONTAINER DELIVERY POINT
/2/     RCVCD (1 = TRETC (P = 13  $                 + RECEIVE CONTAINER
                          1 = RTURN (P = 0),        + SEND CONTAINER
                              *TRANS))              + TO NEXT ACCT
```

Figure II-3. Container Delivery Point Node

ADLVR and CDP are set to be executed at every rate/level time step to set the return address for a shipment and calculate the receipt rate into the account. Subnode 2 is only executed when a container is delivered to the account. RCVCD unloads the shipments for the account and adds them to the flow level. In this example, if the container contains other shipments it is sent back to transportation to be delivered to the next account in a route.

2. CCP Without Discrete Transportation

The containerization modules have been used to represent the creation of discrete container loads to measure container utilization and consolidation delays, to assess losses of container loads, and then immediately translate the shipments back to a flow of materiel. This technique eliminates the representation of each shipment in each container load for the simulated travel time, thereby cutting computer core storage requirements. However, this also assumes that no significant overloads will occur in the transportation network and that containers will be delivered after a specified delay time.

Figure II-4 shows a description of a CCP node without a discrete transportation simulation. The first subnode is executed every time step, translating the incoming flow rates to discrete shipments for consolidation in RCVCP and determine the outgoing flow rate from the level of materiel that has been containerized and will not be lost. Subnode 2 represents CCP operations with the only change occurring in PS4 of verb CCP. Here, instead of a loaded container being sent to transportation, losses on two legs of the journey are determined, and then the module CDLVL unpacks the container and translates the discrete shipments back to a level of materiel which will determine the outgoing rate.

3. Full Container Assignment at Depots

In the Army logistic system certain materiel is sent directly to a customer, based on the volume of demands, rather than through the CCP for consolidation with other materiel. The containerization module family contains modules which represent this situation. Figure II-5 shows partial node descriptions which represent full container shipments from a CONUS

```
+
+    CCPE - CONSOLIDATION AND CONTAINERIZATION POINT.  CONTAINER LOADS ARE
+           BUILT, LOSSES ARE ASSESSED, AND SHIPMENTS ARE CONVERTED BACK
+           TO FLOW
+
CCPE.     RLNOD     (P = *CCPE, 81.),        + RL NODE NAME
          RCVCP     (P = 1),                 + RECEIVE FLOW AT CCP
          SNDCP     (P = 1),                 + SEND FLOW FROM CCP
          *RLCTR                             + NEXT NODE
+
+    SUBNODE 2 - CCP OPERATIONS
+
/2/  CCP (1 = CASIN (P = 0)  $
          2 = DASIN (P = 0)  $
          3 = CLOAD          $
          4 = CLOSS (P = 2, 1, *CONUS, 2, *WRSL), + ASSESS LOSSES
              CDLVL          $                    + CONTAINER DELIVERY LEVEL
          5 = RTURN (P = 0), *CCPE .2)            + RESCHEDULE CCPE .2
     $
```

Figure II-4. CCP Node Without Discrete Transportation

```
+    FULL CONTAINER ASSIGNMENT AND SHIPMENT

+    THRU TRANSPORTATION
+
     CICPD     (1 = FCASN     (P = 0    $    + FULL CONTR ASSIGN.
                               1 = FCLOD (1 = TRETC (P = 56  $
                                                     1 = RTURN (P = 0),
                                                        *TRANS))),
                                                + SET CLASS DENSITY FOR CCP

               MDENS),

+

+    FULL CONTAINER ASSIGNMENT, NO TRANSPORATION

+

     CICPD     (1 = FCASN     (P = 0, 0, 0  $
                               1 = FCLOD (1 = CDLVL))),

     MDENS
```

Figure II-5. Full Container Assignment at Depot Examples

depot. The first example utilizes discrete transportation to send the loaded containers. The second example merely measures container utilization and translates the discrete shipments back to a flow of materiel. Losses could also have been assessed here by placing CLOSS in front of CDLVL in parameter slot 1 of FCLOD.

CHAPTER III
CONTAINERIZATION DATA STRUCTURE

A. DATASET DESCRIPTIONS

One common deck and several PDS datasets are defined for the containerization module family. The common deck is utilized throughout the modules to transmit current information. The datasets are keyed on by the modules for characteristics of containers and policy parameters and can be specified at model input time. Table III-1 lists the different datasets by name. Table III-2 describes the variables in the common block /CONCOM/. Tables III-3 through III-19 describe the elements of the containerization. The containerization module family is designed so the number of types and characteristics of containers used is flexible and not specified until model input time. The only maximum on the variety of containers used and the number of customers serviced is the core storage available for all permanent datasets. Aggregate customers in a model may be split up into individual customers at a containerization by using the AGGSF dataset which splits the flow of materiel coming into the containerization point.

B. SHIPMENT REPRESENTATION

Each flow rate that enters the CCP is translated into a discrete shipment and entered into a stack of shipments waiting for consolidation. Each account serviced by a CCP has a stack for each priority of materiel being sent to it. Thus, all classes of high priority materiel for an account are entered into the same stack. The pushdown stack mechanism for IZHOLD is utilized within the module logic. Each entry in a stack has the following form:

Class	Age	Density	Cube of Materiel	Pointer
CPAK2	CPAK3			

TABLE III-1. CONTAINERIZATION DATASETS

NAME	PDS TABLE	DESCRIPTION
/CONCOM/		CONTAINERIZATION COMMON BLOCK
ACCLOC	ACCLOC	ACCOUNT LOCATION INFORMATION
AGGACC	CCPTAB	ACCOUNT FACTORS FOR INDIVIDUAL ACCOUNTS IN AN AGGREGATE ACCOUNT
AGGSF	CCPTAB	SPLITTING FACTORS FOR ACCOUNTS IN AGGREGATE
CCPPOL	CCPPOL	CONTAINERIZATION AND CONSOLIDATION POINT POLICIES
CCPSTS	CCPTAB	CCP STATISTICS
CDLEVL	CDPTAB	CONTAINER DELIVERY LEVELS
CDSTAT	CDPTAB	CONTAINER DELIVERY STATISTICS
CLOSS	NODPRI	CONTAINER LOSS FACTORS
CLUSTR	CLUSTR	LIST OF ACCOUNTS IN CLUSTER
CONPOL	CONTNR	CONTAINER POLICY FOR A NODE
CONSTS	CONTNR	CONTAINER LOADING STATISTICS
CONTNR	CONTNR	CONTAINER CHARACTERISTICS
CONTYP	CCPPOL	TYPES OF CONTAINERS AT A NODE
DPOFAC	NODPRI	DEPOT FACTORS FOR FULL CONTAINER LOADING
DPOLVL	DPOLVL	LEVEL OF MATERIEL TO BE SENT TO ACCOUNT IN FULL CONTAINERS BY CLASS
DROPPT	CLUSTR	LIST OF ACCOUNTS SERVICED BY A DROP POINT
PRIORT	CCPPOL	LIST OF PRIORITIES EXPECTED FOR CONTAINERIZATION

TABLE III-2. LABELED COMMON /CONCOM/

VARIABLE	DESCRIPTION
CLSTRI	INDEX TO CLUSTER CURRENTLY BEING CONSIDERED
PRIORI	INDEX TO PRIORITY CURRENTLY BEING CONSIDERED
CURAGE	CURRENT AGE (MOD 100) USED TO CHECK SHIPMENT AGE
NFACCT	FIRST ACCOUNT TO START AT IN CLUSTER FOR ASSIGNMENT AND LOADING PROCEDURE
NUMACC	NUMBER OF ACCOUNTS TO INCLUDE IN A LOAD
TYPMIN	TYPE OF CONTAINER FOR MINIMUM DIFFERENCE FROM MIN CUBE
TYPMAX	TYPE OF CONTAINER FOR MIN DIFFERENCE FROM MAX CUBE
TYPLRG	TYPE OF CONTAINER FOR LARGEST AMOUNT LESS THAN OR EQUAL TO CUBE
KDROPT	KEY TO CLUSTER/DROP POINT LOADING (0-CLUSTER, 1-DROP POINT, 2-EITHER)
LSEND	KEY TO SEND CONTAINER LOAD DUE TO AGE (0-NO, 1-YES)
LDGO	KEY TO LOAD CONTAINER AND GO (0-DON'T LOAD, 1-LOAD)
LRECYC	KEY TO RECYCLE CONTAINER ASSIGNMENT FOR MORE MATERIEL (0-DON'T, 1-RECYCLE)
CPAK1	PACKING FACTOR FOR CLASS IN CONTAINER SHIPMENT STACK ENTRY
CPAK2	PACKING FACTOR FOR CLASS IN CCP STACK ENTRY
CPAK3	PACKING FACTOR FOR AGE IN CCP STACK ENTRY
CONTYP	CURRENT CONTAINER TYPE UNDER CONSIDERATION
ACCNT	CURRENT ACCOUNT BEING ASSIGNED SHIPMENTS
CLASS	CURRENT CLASS OF MATERIEL BEING CONSIDERED

TABLE III-3. ACCLOC DATASET

ACCLOC (ACCOUNT) - ACCOUNT LOCATION INFORMATION

NAME	ELEMENT NO.	DESCRIPTION
TERM	1	TRANSPORTATION TERMINAL SERVING ACCOUNT
CDPTR	2	RETURN SHIPMENT POINTER TO CDP
CLUSTR	3	CLUSTER THAT ACCOUNT IS IN

TABLE III-4. AGGACC DATASET

AGGACC (NODE, ACCOUNT, INDEX *100+PRI)* - ACCOUNT FACTORS FOR INDIVIDUAL ACCOUNTS IN AN AGGREGATE ACCOUNT

NAME	ELEMENT NO.	DESCRIPTION
AGGCUB	1	CUBE OF MATERIEL CURRENTLY IN AGGREGATE STACK RELATED TO THIS ACCOUNT
AGGAGE	2	TIME THAT OLDEST MATERIEL FOR THIS ACCOUNT WAS PLACED IN STACK

*THE INDEX IS OBTAINED FROM DATASET CLUSTR OR DROPPT WHERE AGGREGATE ACCOUNTS ARE LISTED AS IJ.KL FOR AGGREGATE ACCOUNT IJ, INDIVIDUAL ACCOUNT KL.

TABLE III-5. AGGSF DATASET

AGGSF (NODE, ACCOUNT, 0) - SPLITTING FACTORS FOR ACCOUNTS IN AGGREGATE

NAME	ELEMENT NO.	DESCRIPTION
AGSF1	1	FRACTION OF AGGREGATE ACCOUNT FLOW RELATED TO INDIVIDUAL ACCOUNT 1
AGSF2	2	FRACTION OF AGGREGATE ACCOUNT FLOW RELATED TO INDIVIDUAL ACCOUNT 2
	.	
	.	
	.	
AGSF20	20	FRACTION OF AGGREGATE ACCOUNT FLOW RELATED TO INDIVIDUAL ACCOUNT 20

TABLE III-6. CCPPOL DATASET

CCPPOL (NODE) - CCP POLICIES

NAME	ELEMENT NO.	DESCRIPTION
NCONSN	1	MAX NUMBER OF CONSIGNEES IN A CONTAINER (NOT APPLICABLE TO DROP POINTS)
CURAGE	2	CURRENT AGE
LODPOL	3	KEY - SINGLE ACCOUNT CONTAINER LOAD GREATER THAN MIN ECONOMICAL WEIGHT TO BE SENT 0-NO, 1-YES

TABLE III-7. CCPSTS DATASET

CCPSTS (NODE, ACCOUNT, PRIORITY) - CCP STATISTICS FOR EACH ACCOUNT

NAME	ELEMENT NO.	DESCRIPTION
PTR-HT	1	POINTER TO STACK CONTAINING SHIPMENTS
CUBE	2	CUBE OF MATERIEL IN STACK
WEIGHT	3	WEIGHT OF MATERIEL IN STACK
SI/CUB	4	STAT IND - CUBE OF MATERIEL IN STACK (3)
SI/WGT	5	STAT IND - WEIGHT OF MATERIEL IN STACK (3)
SI/HTM	6	STAT IND - HOLD TIME OF MATERIEL IN STACK (3)
CCPSTS (NODE, 0, PRIORITY) - CCP STATISTICS FOR ALL ACCOUNTS		
PTR-HT	1	MAXIMUM HOLD TIME
CUBE	2	CUBE OF MATERIEL IN CCP
WEIGHT	3	WEIGHT OF MATERIEL IN CCP
SI/CUB	4	STAT IND - CUBE OF MATERIEL IN CCP (3)
SI/WGT	5	STAT IND - WEIGHT OF MATERIEL IN CCP (3)
SI/HTM	6	STAT IND - AVERAGE HOLD TIME OF SHIPMENTS (3)

TABLE III-8. CDLEVL DATASET

CDLEVL (ACCOUNT, CLASS) - CONTAINER DELIVERY LEVELS

NAME	ELEMENT NO.	DESCRIPTION
LEVEL	1	CURRENT LEVEL OF MATERIEL WHICH HAS COME INTO CDP BUT HAS NOT BEEN TRANSFERRED TO RATE YET

TABLE III-9. CDSTAT DATASET

CDSTAT (ACCOUNT, PRIORITY) - CONTAINER DELIVERY STATISTICS

NAME	ELEMENT NO.	DESCRIPTION
SI/NCR	1	STAT IND - NUMBER OF CONTAINERS RECEIVED (1)
SI/CTT	2	STAT IND - TRAVEL TIME OF CONTAINER (3)
SI/CUB	3	STAT IND - CUBE OF MATERIEL RECEIVED (3)
SI/WGT	4	STAT IND - WEIGHT OF MATERIEL RECEIVED (3)
SI/NCD	5	STAT IND - NUMBER OF CONTAINERS DIVERTED (1)
SI/WTL	6	STAT IND - WEIGHT OF MATERIEL LOST (1)

TABLE III-10. CLOSS DATASET

CLOSS (NODE, PRIORITY) - CONTAINER LOSS FACTORS

NAME	ELEMENT NO.	DESCRIPTION
LDI1	1	LOSS PERCENTAGE, LEG 1
SI/CL1	2	STAT IND - NUMBER OF CONTAINERS LOST, LEG 1 (1)
SI/WL1	3	STAT IND - WEIGHT OF MATERIEL LOST, LEG 1 (1)
LDI2	4	LOSS PERCENTAGE, LEG 2
SI/CL2	5	STAT IND - NUMBER OF CONTAINERS LOST, LEG 2 (1)
SI/WL2	6	STAT IND - WEIGHT OF MATERIEL LOST, LEG 2 (1)
LDI3	7	LOSS PERCENTAGE, LEG 3
SI/CL3	8	STAT IND - NUMBER OF CONTAINERS LOST, LEG 3 (1)
SI/WL3	9	STAT IND - WEIGHT OF MATERIEL LOST, LEG 3 (1)

TABLE III-11. CLUSTR DATASET

CLUSTR (NODE, CLUSTER) - LIST OF ACCOUNTS IN A CLUSTER

NAME	ELEMENT NO.	DESCRIPTION
ACCT1	1	FIRST ACCOUNT IN ROUTE
ACCT2	2	SECOND ACCOUNT IN ROUTE
.		
.		
.		
ACCTn	n	LAST ACCOUNT IN ROUTE
	CLUSTR (NODE, 0)	- LIST OF CLUSTERS AT THIS NODE
ACCT1	1	FIRST CLUSTER NUMBER
ACCT2	2	SECOND CLUSTER NUMBER
.		
.		
.		
ACCTn	n	LAST CLUSTER NUMBER

TABLE III-12. CONPOL DATASET

CONPOL (NODE, CONTAINER TYPE) - CONTAINER POLICY AT A NODE

NAME	ELEMENT NO.	DESCRIPTION
NCONT	1	NUMBER OF CONTAINERS AVAILABLE
HIPRI	2	HIGHEST PRIORITY ALLOWED IN THIS CONTAINER TYPE
LOPRI	3	LOWEST PRIORITY ALLOWED IN THIS CONTAINER TYPE
MINWGT	4	MIN WEIGHT TO SHIP CONTAINER ECONOMICALLY
KDROPT	5	CONTAINER SENT TO ACCOUNT (0), DROP POINT (1), OR BOTH (2)

TABLE III-13. CONSTS DATASET

CONSTS (NODE, CONTAINER TYPE) - CONTAINER STATISTICS

NAME	ELEMENT NO.	DESCRIPTION
SI/NCS	1	STAT IND - NUMBER OF CONTAINERS SHIPPED (1)
SI/PCU	2	STAT IND - PERCENT CUBE UTILIZATION (3)
SI/CUB	3	STAT IND - TOTAL CUBE SHIPPED (1)
SI/WGT	4	STAT IND - TOTAL WEIGHT SHIPPED (1)
SI/NCN	5	STAT IND - AVERAGE NUMBER OF CONSIGNEES (3)
SI/ICS	6	STAT IND - NUMBER OF CONTAINERS SENT TO ONLY 1 CONSIGNEE (1)
SI/NUC	7	STAT IND - NUMBER OF UNECONOMICAL CONTAINERS SENT (1)
SI/NC	8	STAT IND - NUMBER OF CONTAINERS AVAILABLE (3)

TABLE III-14. CONTNR DATASET

CONTNR (0, CONTAINER TYPE) - CONTAINER CHARACTERISTICS

NAME	ELEMENT NO.	DESCRIPTION
MAXCUB	1	MAXIMUM CUBE CAPACITY OF CONTAINER
MAXWGT	2	MAXIMUM WEIGHT CAPACITY OF CONTAINER
WTN060	3	MINIMUM CUBE WITH WHICH CONTAINER WILL NOT GO

TABLE III-15. CONTYP DATASET

CONTYP (NODE) - CONTAINER TYPES AVAILABLE AT THIS NODE

NAME	ELEMENT NO.	DESCRIPTION
CONTP1	1	FIRST CONTAINER TYPE
CONTP2	2	SECOND CONTAINER TYPE
.		
.		
.		
CONTPn	n	nTH CONTAINER TYPE

TABLE III-16. DPOFAC DATASET

DPOFAC (NODE, PRIORITY) - DEPOT FACTORS FOR FULL CONTAINER LOADING

NAME	ELEMENT NO.	DESCRIPTION
MAXHLD	1	MAXIMUM HOLD TIME TO FILL CONTAINERS
OPTLOD	2	OPTIMUM LOAD POINT OF CONTAINER (% OF CAPACITY)

TABLE III-17. DPOLVL DATASET

DPOLVL (NODE, ACCOUNT*, CLASS) - LEVEL OF MATERIEL TO BE SENT TO ACCOUNT IN FULL CONTAINERS BY CLASS

NAME	ELEMENT NO.	DESCRIPTION
LVMC1	1	LEVEL OF MATERIEL IN MANAGEMENT CLASS 1
LVMC2	2	LEVEL OF MATERIEL IN MANAGEMENT CLASS 2
LVMC3		
.		
.		
.		
LVMCm	m	LEVEL OF MATERIEL IN MANAGEMENT CLASS m
AGMC1	m+1	AGE OF MATERIEL IN MANAGEMENT CLASS 1
.	.	
.	.	
.	.	
AGMCm	2m	AGE OF MATERIEL IN MANAGEMENT CLASS m

*IF AGGREGATED ACCOUNT, WILL HAVE ACCOUNT INDEX IN TWO DIGITS TO RIGHT OF DECIMAL POINT BASED ON AGGSF.

TABLE III-18. DROPPT DATASET

DROPPT (NODE, CLUSTER) - LIST OF ACCOUNTS SERVICED BY A DROP POINT IN CLUSTER

NAME	ELEMENT NO.	DESCRIPTION
ACCT1	1	FIRST ACCOUNT IN CLUSTER (DROP POINT)
ACCT2	2	SECOND ACCOUNT IN CLUSTER
.	.	
.	.	
.	.	
ACCTn	n	LAST ACCOUNT IN CLUSTER

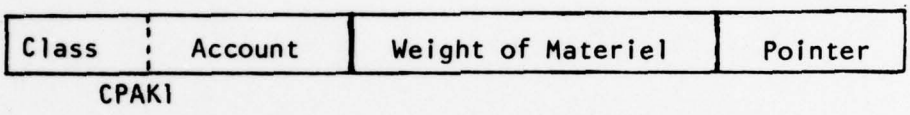
TABLE III-19. PRIORT DATASET

PRIORT (NODE) - LIST OF PRIORITIES EXPECTED

NAME	ELEMENT NO.	DESCRIPTION
PRIOR1	1	HIGHEST PRIORITY FOR FIRST STACK FOR ANY ACCOUNT AT CCP
PRIOR2	2	HIGHEST PRIORITY FOR SECOND STACK AT CCP
.		
.		
.		
PRIORn	n	LOWEST PRIORITY FOR ANY STACK AT CCP

The CCP logic can then scan the pushdown stacks for the proper amount and age of materiel to be loaded in a container for an account or a cluster of accounts. The stacks are handled in a first in, first out (FIFO) basis so that the oldest shipments are sent first.

Each container that is loaded and sent through the system is represented by a set of attributes and a pushdown stack of shipments that are in the container. In this manner, the contents of every container are known and delivery of the container allows the delivery level of each class of materiel to be properly affected. The definition of container attributes is given in Table III-20. Attribute 8 has a pointer to the pushdown stack where the shipments in the container are stored. The IZHOLD structure is also used for these stacks with the following form for a stack entry:



C. STATISTICS CODES

The statistics collected in the containerization family of modules are defined in Table III-21. The type codes utilized are defined in Table III-22. These codes correspond to the statistics structure utilized in the MAWLOGS transportation module families and the item based supply and maintenance families.

TABLE III-20. CONTAINER ATTRIBUTE SET

ATTRIB NO.	DESCRIPTION
1	TIME
2	SOURCE TERMINAL * 1000 + SINK TERMINAL
3	FINAL DEST. TERM + 1000. * RETURN SHIPMENT POINTER
4	NOT USED
5	CONTAINER TYPE CODE (OR KEY TO SHOW CONTAINER LOST (-99))
6	PRIORITY
7	SHIPMENT ROUTING
8	POINTER TO STACK WITH CONTENTS OF CONTAINER * 1000. + KEY TO ACCOUNT/DROP POINT DELIVERY
9	TIME CONTAINER SHIPPED
10	CUBE OF LOADED CONTAINER
11	WEIGHT OF LOADED CONTAINER

TABLE III-21. CONTAINERIZATION STATISTICS

TYPE CODE	DESCRIPTION	RESOURCE ID		STAT. INDICATOR STORAGE LOCATION
		PRIM	SEC	
14314	CUBE OF MATERIEL IN STACK	ACCOUNT, PRIORITY		CCPSTS ( ).4
24314	WEIGHT OF MATERIEL IN STACK	ACCOUNT, PRIORITY		CCPSTS ( ).5
144334	HOLD TIME OF MATERIEL IN STACK	ACCOUNT, PRIORITY		CCPSTS ( ).6
136334	NUMBER OF CONTAINERS SHIPPED	CONTAINER TYPE, 0		CONSTS ( ).1
66334	PERCENT CUBE UTILIZATION	CONTAINER TYPE, 0		CONSTS ( ).2
16334	TOTAL CUBE SHIPPED	CONTAINER TYPE, 0		CONSTS ( ).3
26334	TOTAL WEIGHT SHIPPED	CONTAINER TYPE, 0		CONSTS ( ).4
56334	NUMBER OF CONSIGNEES	CONTAINER TYPE, 0		CONSTS ( ).5
436334	NUMBER OF 1 CONSIGNEE CONTAINERS	CONTAINER TYPE, 0		CONSTS ( ).6
336334	NUMBER OF UNECONOMICAL CONTAINERS	CONTAINER TYPE, 0		CONSTS ( ).7
536314	NUMBER OF CONTAINERS AVAILABLE	CONTAINER TYPE, 0		CONSTS ( ).8
135334	NUMBER OF CONTAINERS RECEIVED	ACCOUNT, PRI		CDSTAT ( ).1
245334	TRAVEL TIME OF CONTAINER	ACCOUNT, PRI		CDSTAT ( ).2
15334	CUBE OF MATERIEL RECEIVED	ACCOUNT, PRI		CDSTAT ( ).3
25334	WEIGHT OF MATERIEL RECEIVED	ACCOUNT, PRI		CDSTAT ( ).4
235334	NUMBER OF CONTAINERS DIVERTED	ACCOUNT, PRI		CDSTAT ( ).5
625334	WEIGHT OF MATERIEL LOST	ACCOUNT, PRI		CDSTAT ( ).6
636334	NUMBER OF CONTAINERS LOST, 1ST LEG	CONTAINER TYPE, 1		CLOSS ( ).2
626334	WEIGHT OF MATERIEL LOST, 1ST LEG	CONTAINER TYPE, 1		CLOSS ( ).3
636334	NUMBER OF CONTAINERS LOST, 2ND LEG	CONTAINER TYPE, 2		CLOSS ( ).5
626334	WEIGHT OF MATERIEL LOST, 2ND LEG	CONTAINER TYPE, 2		CLOSS ( ).6
636334	NUMBER OF CONTAINERS LOST, 3RD LEG	CONTAINER TYPE, 3		CLOSS ( ).8
626334	WEIGHT OF MATERIEL LOST, 3RD LEG	CONTAINER TYPE, 3		CLOSS ( ).9

0 COORDINATE VALUE FOR ACCOUNT, CONTAINER TYPE, OR PRIORITY INDICATES AGGREGATE DATA (I.E., ALL ACCOUNTS)

TABLE III-22. STATISTICS TYPE CODES (fedcba)

a = 4 - TRANSPORTATION FUNCTION

b = 1 - STANDARD DEFINITION (SEE VOL II)

2

3

c = 3 - TRANSPORTATION FAMILY CODE - FLOW CONTAINERIZATION

d, TRANSPORTATION ENTITY

= 4, CONTAINER SHIPMENT POINT

= 5, CONTAINER RECEIPT POINT

= 6, CONTAINERS

e, UNIQUE STATISTIC DESCRIPTION CODE BY ENTITY

= 1, CUBE OF MATERIEL

= 2, WEIGHT OF MATERIEL

= 3, NUMBER OF CONTAINERS

= 4, TIME

= 5, NUMBER OF CONSIGNEES

= 6, PERCENT CUBE UTILIZATION

e = 2, f = 6, WEIGHT OF MATERIEL LOST

e = 3, f = 1, CONTAINERS SHIPPED/RECEIVED

= 2, CONTAINERS DIVERTED

= 3, UNECONOMICAL CONTAINERS

= 4, CONTAINERS SENT TO ONLY 1 CONSIGNEE

= 5, AVAILABLE FOR ASSIGNMENT

= 6, CONTAINERS LOST

e = 4, f = 1, HOLD TIME OF MATERIEL

f = 2, TRAVEL TIME OF CONTAINER

e = 2, f = 6, WEIGHT LOST

CHAPTER IV
CONTAINERIZATION VERBS

THIS PAGE LEFT INTENTIONALLY BLANK

ADLVR (ACCNAM, IZSTOC)

General Description

This module identifies an account delivery point, that is it identifies in its parameter slot the location in the model logic where control will be tranfered when materiel is to be delivered to the account. The account name is entered in the argument as P = *ACCNAM and this is used to access the PDS dataset ACCLOC so that a return flow pointer may be entered in element 2. The argument IZSTOC allows a delay time to be entered for later use. The return flow pointer points to a stack entry that is equivalent to an RNOD entry in word 1 but contains the account index number in word 2. It is a single entry stack. ADLVR is designed to be used with rates and levels accounts and containerized delivery. It is envisioned that the return flow entry will be permanent and read only by the verb RFLON when a container is delivered to a container delivery point. Thus ADLVR should be executed only once when the R and L linkage is being defined.

Assembler Inputs

Arguments

ACCNAM - name of the account to which container deliveries are to be made

IZSTOC - index to a delay time for final container delivery

Parameter Slots

1 - destination node for delivery to an account

Examples

The module ADLVR should be inserted at the beginning of a demand generator node to identify a container delivery point.

```
ANODE.  RLNOD (P = *ANODE, 31.),      + RL NODE NAME
        ADLVR (P = *ANODE, 0     $
               1 = *ANODE .2), . . .  + DELIVERY POINT
        /2/ RCVCD  $
```

Statistics Collected None.

GASP Files Used. None.

Permanent Attributes Accessed

PDS Datasets:

ACCLOC (account index).2 - return shipment pointer

Verb Inputs

From Calling Program None.

From ZPLAEN

Common/ZSTACK/:

IZREF - pointer to entry in IZHOLD stack which contains the return shipment information

Verb Outputs

To ZPLAEN

Common/ZSTACK/:

IZK - account index number

IZKC - IZNODE + IP12 * IZVERB + IP24 * IZSTOC

IZREF - 0

To PERMDS

In ACCLOC (account index).2 -

IZREF - return shipment pointer

Programs Called

Verbs. None.

Other. RLNIX, ZPLAEN, PERMDS

Input/Output Files Used. None.

CASIN (KNCONT)

General Description

This module attempts to assign one container load for a cluster of accounts at a CCP. It must be given the following information -

CLSTRI - the cluster to work with

PRIORI - the priority of shipments to consider

NFACCT - the first account in the cluster at which the assignment will start

The weight of shipments at each account in a cluster is checked in turn and all possible container types are scanned to determine the best type, if any, to use. The key LDGO is set to 1 if a container load is assigned. The module CLOAD may then be utilized to load the proper shipments in a container. 80 percent of a container capacity is considered a full (optimum) load for assignment purposes. CLOAD will load up to 100 percent if materiel is available.

Any of the following conditions will cause a container load to be assigned ---

1. If the volume of shipments for the first account checked is .GT. The min. cube (i.e., economical fill volume) for a container type, then a container will be sent to that account only.
2. If the volume of shipments for one account is .GT. The max cube for all container types, then the largest container type will be assigned. If this is true for more than one account and the max hold time has been reached for any materiel, the largest type is assigned. This attempts to minimize the number of consignees.
3. If the max number of consignees has been checked an the max hold time for shipments at one of those accounts has been reached, then the smallest container type that will hold the shipments will be assigned. An uneconomical container will not be sent if a smaller type is available.

It should be noted that the loading factor here is called cube. The factor can be considered as weight or cube depending on the input data and report labels. The important idea is that only one loading factor is used, not a combination of both of them.

Assembler Inputs

Arguments

KNCONT - Key to number of containers available -

0 - not constrained

1 - containers constrained

Parameter Slots. None

Examples

This module is used to assign shipments to a container at a CCP for accounts in a cluster routing.

```
CCP (1 = CASIN (P = 0)  $  + ASSIGN LOAD
     3 = CLOAD)
```

Statistics Collected. None.

GASP Files Used. None.

Permanent Attributes Accessed

PDS Datasets:

CLUSTR (node, cluster) - list of accounts in the cluster

CCPSTS (node, 0, priority).1 - maximum holdtime for materiel

CCPPOL (node, 0, priority) -

1 - maximum number of consignees in a container

2 - current time to check age of shipments

3 - loading policy for single consignee shipments

AGGACC (node, account, index and priority) -

1 - cube of materiel for this account in an aggregate

2 - age of materiel

CCPSTS (node, account, priority) -

1 - pointer to stack containing shipments for this account

2 - cube of the materiel in the stack

CONTYP (node) - list of available container types

CONTNR (0, containter type) -

1 - maximum cube capacity of container

3 - minimum cube required to ship container

CONPOL (node, container type) -

1 - number of containers available

2 - highest priority allowed in this container type

3 - lowest priority allowed in this container type

4 - minimum cube to ship container economically

5 - container sent to account (0), drop point (1), or either (2)

Verb Inputs

From Calling Program

Common/CONCOM/:

CLSTRI - the cluster to be considered

PRIORI - the priority of shipments to consider

NFACCT - the first account in the cluster at which the assignment will start

Verb Outputs

To Calling Program

Common/CONCOM/:

LDGO - key to indicate whether load is available (0 - no, 1 - yes)

LRECYC - key to indicate whether another assignment should be attempted for this cluster (0 - no, 1 - yes)

Common/VRBGSP/:

ATRIB(5) - types of container to be used for loading

Programs Called

Verbs. None.

Other. None.

Input/Output Files Used. None.

(CASIN-3)

THIS PAGE LEFT INTENTIONALLY BLANK

CCP

General Description

This is the control module to simulate the operations of a containerization and consolidation point (CCP). It calls various parameter slots to handle assignment, loading, and shipping of containers to individual accounts, drop points, or clusters of accounts on a route.

This verb must be scheduled exogenously to start the initial containerization. Thereafter it will reschedule itself at a fixed interval (obtained from ATRIB(3)). It will keep track of the current age at this node by storing it in CCPPOL (IZNODE).2 and updating it when CCP is called. CURAGE is stored modulo 100 and is used to check the age of shipments in account stacks. Modulo 100 is used since the age of a shipment is packed in the first word of the stack entry.

If the CCP recycle time is shorter than the rate/level time step, the incoming flow is split into smaller pieces and only the current shipments considered. At the end of CCP operations, the next periods shipments are activated for consideration.

Assembler Inputs

Arguments. None.

Parameter Slots

1 - assign a container load to an account at a cluster

2 - assign a container load to a drop point in a cluster

3 - load a container

4 - send a loaded container (RFLON or TRETC)

5 - R-delay to reschedule this module

Examples

This is the main verb at a CCP node. Cluster assignment or drop point assignment is chosen by contents of PS1 and PS2.

```
/2/ CCP (1 = CASIN (P = 0)  $
```

2 = DASIN (P = 0) $
3 = CLOAD $
4 = CDLVL $ + COULD SEND TO TRANSPORT.
5 = RTURN (P = 0), * CCPE.2)

Statistics Collected

PDS dataset CCPSTS (node, account, priority) -
4 - cube of materiel in CCP holding stack for account
5 - weight of materiel in CCP holding stack for account
PDS dataset CCPSTS (node, 0, priority) -
4 - cube of all materiel in CCP
5 - weight of all materiel in CCP

GASP Files Used. None.

Permanent Attributes Accessed

PDS Datasets:
CCPPOL (node).2 - current age, modulo 100.
PRIORT (node) - list of priorities expected
CLUSTR (node, cluster) - list of accounts in cluster
DROPPT (node, cluster) - list of accounts serviced by a drop point
CCPSTS (node, account, priority) -
1 - pointer to shipments stack
2 - cube of materiel in stack
3 - weight of materiel in stack
4 - STAT IND - cube in stack
5 - STAT IND - Wgt. in stack
AGGSF (node, account) - aggregate node splitting factors
AGGACC (node, account, index * 100 + pri) -
1 - cube of materiel
2 - age of materiel

Verb Inputs

From Calling Program

Common/VRBGSP/:
ATRIB(5) - time delay to schedule next CCP operations.

From PS1

Common/CONCOM/:

LDGO - flag to specify whether load is available

LRECYC - flag to specify whether another assignment should be attempted

Common/VRBGSP/:

ATRIB(5) - type of container to be used for loading

From PS2

Common/CONCOM/:

LDGO - load availability flag

LRECYC - flag for another assignment attempt

Common/VRBGSP/:

ATRIB(5) - type of container to be used for loading

From PS3

Common/VRBGSP/:

ATRIB - attributes of container to be shipped

From PS4. None.

From PS5. None.

Verb Outputs

To PS1

Common/CONCOM/:

CLSTRI - cluster index

PRIORI - priority

NFACCT - first account for assignment

To PS2

Common/CONCOM/:

CLSTRI - cluster index

PRIORI - priority

NFACCT = 1, start at first account (drop point)

To PS3

Common/VRBGSP/:

ATRIB(5) - contains type to be used for loading

Common/CONCOM/:

CLSTRI - cluster index

PRIORI - priority

NFACCT - first account for loading

To PS4

Common/VRBGSP/:

ATRIB - attributes of loaded container

To PS5

Common/VRBGSP/:

ATRIB - attributes as input to CCP with event time incremented by ATRIB(3)

To Calling Program. None.

Programs Called

Verbs. None

Other. IDSLON

Input/Output Files Used. None.

CDLVL

General Description

This module removes the contents of a container and enters the value in the appropriate CDLEVL dataset. All shipments are removed and placed in CDLEVL (ACCNT, CLASS).1. This dataset will be cleared and the contents changed to rates by module SNDCP or CDP.

If the container has been lost, then the contents are not transferred and are therefore lost to the system.

Assembler Inputs

Arguments. None

Parameter Slots

1 - Dispose of the empty container

Examples

This module may be used at a CCP node after a container load has been formed.

```
CCP (1 = CASIN (P = 0)  $
     3 = CLOAD          $
     4 = CDLVL          $  + TRANSLATE BACK TO FLOW
```

Statistics Collected

PDS Dataset CDSTAT (account, priority) -

1 - number of containers sent out

2 - weight of materiel sent out

GASP Files Used. None.

Permanent Attributes Accessed

PDS Datasets:

CDLEVL (account, class).1 - weight of materiel

If CDLEVL dataset does not exist, this module creates it.

Verb Inputs

From Calling Program

Common/VRBGSP/:

ATRIB - attributes of container

5 - container type code or -99 if container has been lost

6 - priority

8 - 1000. *pointer to stack with contents of container

9 - time container shipped

10 - total cube of shipments in container

11 - total weight of shipments in container

IZHOLD - pushdown stack with container shipments

From PSl. None.

Verb Outputs

To PSl

Common/VRBGSP/:

ATRIB(5) - type of container

To Calling Program. None.

Programs Called

Verbs. None.

Other. ZREMEN.

Input/Output Files Used. None.

CDP (IRR)

General Description

This container delivery point module translates the available levels for each account and class to rates. This is the point of change from discrete shipments to continuous rates. This module should be executed at every continuous time step. The datasets CDLEVL for each account and class at this node are looped through. The receipt rate to be changed is pointed to by FCRAT (ACCT, CLASS). IRR + ND + 2. The incoming rate at the account may be altered by a delay time. This would be especially applicable at accounts served by a drop point. The rates are stored in array DTABLE, the account number and delay time in position 1, the rate in position 2. (DTABLE is equivalenced to IDTABLE.

Assembler Inputs

Arguments

IRR - index to the receipt rate to be used for the materiel received by container. Primary (1), secondary (2), . . .

Parameter Slots. None.

Examples

The verb CDP must be placed in a node so that it is executed every R/L time step. The verb RCVCD should be in a separate subnode that is executed only when a container arrives.

```
NODEA.  RLNOD (P = *NODEA, 35.),
                .
                .
                .
        CDP (P = 1),
        * NODEB  $
```

Statistics Collected. None.

GASP Files Used. None.

Permanent Attributes Accessed

PDS Datasets:

FCRAT (account, class) - flow rate dataset

CDLEVL (account, class).1 - level of materiel received by container

Verb Inputs

From Calling Program. None.

From LRLOOP

RRPTR - receipt rate pointer

INDEX - index of rate link coordinates in RLC

From IRSTK

IXACCT - position in DTABLE of receipt rate for this account

From RLDLY

Value of rate/level delay for receipt rate into account

Verb Outputs

To LRLOOP FCRAT dataset name

To IRSTK Account number

To RLDLY

IXDLY - index to delay time

CLEVL/TSTEP - rate of flow

Programs Called

Verbs. None.

Other. IRSTK, IUNPARK, LRLOOP, RLDLY.

Input/Output Files Used. None.

CLOAD

General Description

This module loads a container with materiel from the account stacks at a CCP. The information needed by the module is the cluster, the priority, the container type of use, the number of accounts to include, a key to indicate whether the container is for tailgate delivery to a cluster or to a drop point. (KDROPT = 0 or 1), and the index of the account to start at.

The attributes of the container are defined and each shipment to be loaded is transferred from a CCP holding stack to the stack of shipments in the container. The module utilizes one factor (called cube here) to load containers. It may be considered as weight if that factor if desired. The loading factor is stored in ATRIB (10) for a container and in the second word of the IZHOLD stack entry. A conversion factor is used to calculate the weight from the cube and place it in ATRIB (11). If weight or cube are used alone in this model, the conversion factor should be set to one. It is packed in IZHOLD (1, IZREF).

Assembler Inputs

Arguments. None.

Parameter Slots. None.

Examples

This module is used after an assignment has been made by CASIN or DASIN at a CCP.

```
CCP (1 = CASIN (P = 0)  $   + ASSIGN LOAD
     3 = CLOAD          $   + LOAD CONTAINER
```

Statistics Collected

PDS Dataset CCPSTS (node, account) -

- 4 - cube of materiel in CCP stack
- 5 - weight of materiel in CCP stack
- 6 - hold time of materiel in CCP stack

PDS dataset CCPSTS (node, 0) - statistics for all accounts

PDS dataset CONSTS (node, container type) -

1 - number of containers shipped

2 - percent cube utilization

4 - total weight shipped

5 - number of consignees

6 - number of containers sent to one consignee only

7 - number of uneconomical containers sent

8 - number of containers available

GASP Files Used. None.

Permanent Attributes Accessed

PDS Datasets:

CCPSTS (node, account) -

1 - pointer to stack containing shipments

2 - cube of materiel in stack

3 - weight of materiel in stack

CLUSTR (node, cluster) - list of accounts in cluster

DROPPT (node, cluster) - list of accounts serviced by a drop point

CCPPOL (node).2 - current time at CCP, modulo 100.

CONTNR (0, container type) -

1 - max cube capacity of container

2 - max weight capacity of container

AGGACC (node, account, index *100 + priority) -

1 - cube of materiel for individual account

2 - age of materiel

ACCLOC (account).2 - return shipment pointer for container delivery

Verb Inputs

From Calling Program

Common/CONCOM/:

CLSTRI - cluster to be loaded for

PRIORI - priority of materiel to be used

NFACCT - first account in cluster to load for

NUMACC - number of accounts to load materiel for

KDROPT - type of loading (0 - tailgate, 1 - drop point)

(CLOAD-2)

Common/VRBGSP/:

ATRIB(5) - type of container

From Zremen. Stack engry for shipment at CCP:

IZHOLD(1, IZREF) - packed identify of shipment,
CLASS * CPAK2 + AGE * CPAK3 + DNSITY

IZHOLD(2, IZREF) - cube of materiel in shipment

Verb Outputs

To ZREMEN. Stack entry for shipment in container:

IZHOLD(1, IZREF) - identity of shipment, CLASS * CPAK1 + ACCNT

IZHOLD(2, IZREF) - weight of materiel in shipment

To Calling Program

Common/VRBGSP/:

ATRIB - attributes of loaded container

1 - TNOW

2 - 0.

3 - 1000. *return shipment pointer to CDP for account

4 - 0.

5 - container type code

6 - priority

7 - 0.

8 - pointer to stack with contents of container + 1000.
*key to account/drop point delivery

9 - time container was shipped

10 - cube of loaded container

11 - weight of loaded container

Programs Called

Verbs. None.

Other. CPLAEN, ZREMEN.

Input/Output Files Used. None.

THIS PAGE LEFT INTENTIONALLY BLANK

CLOSS (NLEGS, IDST1, DNODE1, IDST2, DNODE2, IDST3, DNODE3)

General Description

Container loss module. Checks probability of loss on up to three legs of trip for the container load in ATRIB. If loss is predicted, ATRIB(5) is set to -99 and the materiel level is entered as an increment to the demand rate into the nodes specified in the argument if no loss, the container attributes are unchanged.

This module can be used at a CCP or depot container shipment point to predict losses and create reconstitution demands on up to three different nodes in a system. The module is designed to be used with the continuous supply module families since it interfaces with the discrete shipments in a container and the continuous demand rate into a node.

Assembler Inputs

Arguments

NLEGS - number of legs in the journey

IDST1 - index to demand stack in FCRAT to enter replenishment demand for losses in leg on of journey

DNODE1 - name of node to enter demand at for leg one losses

IDST2,

IDST3 - similar to IDST1 for legs two and three

DNODE2,

DNODE3 - similar to DNODE1 for losses on legs two and three

Parameter Slots. None.

Examples

The CLOSS module can be used at a CCP or a depot full container loading point, generally before module CDLEVL.

```
CCP (1 = CASIN (P = 0)  $
     3 = CLOAD          $
     4 = CLOSS (P = 2, 1, *CONUS, 2, *WRSL),
         CDLVL          $  )
```

(CLOSS-1)

```
or   FCASN (P = 0, 0, 0  $
           1 = FCLOD  (1 = CLOSS (P = 2, 1 *CONUS, 2, *WRSL),
                               CDLVL  ))
```

Statistics Collected

PDS dataset CLOSS (node, priority) -

3*L - 1 - number of containers lost, leg L

3*L - weight of materiel lost, leg L

PDS dataset CDSTAT (account, priority) -

6 - number of containers destined for this account that were lost

GASP Files Used. None.

Permanent Attributes Accessed

PDS dataset FCRAT (account, class) -

IL - the specified demand rate for the account

Verb Inputs

From Calling Program

Common/VRBGSP/:

ATRIB - attributes of loaded container

5 - container type

6 - priority

8 - 1000. *pointer to stack with contents of container

9 - time containers shipped

11 - total weight of shipments in container

From RLNIX. Index of rate/level node name.

From UNIFRM.

SWT - random number for loss probability

From ZREMEN.

Common/ZSTACK/:

IZREF - pointer to next entry in container shipment stack

IZK - weight of materiel in shipment

IZKC - identity of shipment (CLASS * CPAKI + ACCNT)

Verb Outputs

To Calling Program

Common/VRBGSP/:

ATRIB - attributes of container as input unless a loss has occurred, then

5 - -99. to flag loss of this container type

8 - 0 since all shipments in stack have been removed and reconstituted as demands

To ZREMEN

Common/ZSTACK/:

IZREF - pointer to next entry in container shipment stack

To VDDRA1: Arguments -

WSHPMT/TSTEP - demand rate increment due to lost shipment

IDNODE(L) - rate/level node number for demand rate

IDRPTR - index of demand rate is DTABLE to use

Programs Called

Verbs. None.

Other. RLNIX, UDDRA1, VNIFRM, ZREMEN.

Input/Output Files Used. None.

THIS PAGE LEFT INTENTIONALLY BLANK

CONRP

General Description

This module prints reports for containerization operations. All container types used in the model and all clusters specified are listed. Reports for different nodes in the model are triggered by the existence of certain PDS datasets. CCP node reports are printed for every node with a CCPPOL dataset. Full container shipping points at depots are identified by a DPOFAC dataset. Reports on container deliveries are prepared for each node with a CDSTAT dataset. Finally loses in the system are reported.

Example report pages for the different types of reports are included with this module description.

Assembler Inputs

Arguments. None.

Parameter Slots. None.

Examples

The verb CONRP is usually used in the standard ZRPRT node, but can be placed anywhere in the model description where container reports are desired.

Statistics Collected. None.

GASP Files Used. None.

Permanent Attributes Accessed

All containerization characteristics and statistics PDS datasets, specifically CONTNR, ACCLOC, CCPPOL, CLUSTR, DROPPT, PRIORT, CCPSTS, CONSTS, CLOSS, CONTRYP, DPOFAC, CONPOL.

Verb Inputs

From Calling Program. None.

Verb Outputs

To Calling Program. None.

Programs Called

Verbs. None

Other. CLEAR, IDSLON, LIN, PGHDR, PRNTST.

Input/Output Files Used. None.

THIS PAGE LEFT INTENTIONALLY BLANK

Pages IV-27 thru IV-32

Not available

DASIN (KNCONT)

General Description

This module attempts to assign one container load from a cluster to be sent to a drop point. It must be given the following information -

CLSTRI - the cluster to work with

PRIORI - the priority of shipments to consider

NFACCT - the first account in the cluster at which the assignment will start. This is assumed to be 1 since the drop point is listed first in the cluster and there is no restriction on the number of accounts which can be included.

The weight of all shipments for accounts in the cluster is accumulated and all possible container types are scanned to determine the best type, if any, to use. The key LDGO is set to 1 if a container load is assigned. The module CLOAD may then be utilized to load the proper shipments in the container. If more shipments are waiting, the key LRECYC is set to 1 so that the assignment will recycle through this cluster. One of the following conditions will cause a container load to be assigned ---

1. If the weight of shipments is greater than the max weight for all container types, then the largest container type will be assigned.
2. If the max hold time for shipments at one of the accounts has been reached, then the smallest container type that will hold the shipments will be assigned. If this type would be uneconomical, the next smaller type is used.

Assembler Inputs

Arguments

KNCONT - Key to number of containers available -

0 - not constrained

1 - constrained

Parameter Slots. None.

Examples

This module is used to assign shipments to a container at a CCP for accounts in a cluster that are serviced by a single drop point.

```
CCP (2 = DASIN (P = 0) $ + DROP POINT ASSIGNMENT
     3 = CLOAD         )
```

Statistics Collected. None.

GASP Files Used. None.

Permanent Attributes Accessed

PDS Datasets:

DROPPT (node, cluster) - list of accounts serviced by drop point

CCPSTS (node, 0, priority).1 - maximum holdtime for materiel

CCPPOL (node, 0, priority) -

- 1 - maximum number of consignees in a container
- 2 - current time to check age of shipments
- 3 - loading policy for single consignee shipments

AGGACC (node, account, index and priority) -

- 1 - cube of materiel for this account in an aggregate
- 2 - age of materiel

CCPSTS (node, account, priority) -

- 1 - pointer to stack containing shipments for this account
- 2 - cube of the materiel in the stack

CONTYP (node) - list of available container types

CONTNR (0, container type) -

- 1 - maximum cube capacity of container
- 3 - minimum cube required to ship container

CONPOL (node, container type) -

- 1 - number of containers available
- 2 - highest priority allowed in this container type
- 3 - lowest priority allowed in this container type
- 4 - minimum cube to ship container economically
- 5 - container sent to account (0), drop point (1), or either (2)

Verb Inputs

From Calling Program

Common/CONCOM/:

CLSTRI - the cluster to be considered

PRIORI - the priority of shipments to consider

NFACCT - the first account in the cluster at which the assignment will start

Verb Outputs

To Calling Program

Common/CONCOM/:

LDGO - key to indicate whether load is available (0 - no, 1 - yes)

LRECYC - key to indicate whether another assignment should be attempted for this cluster (0 - no, 1 - yes)

Common/VRBGSP/:

ATRIB(5) - type of container to be used for loading

Programs Called

Verbs. None.

Other. None.

Input/Output Files Used. None.

THIS PAGE LEFT INTENTIONALLY BLANK

FCASN (KNCONT, MSW, MSPL)

General Description

This module attempts to assign full container loads at a depot so they may be sent directly, bypassing any consolidation point. The rate in array IFSRI(M) is operated on. This module will work on class or a management class, depending on the values of arguments MSW and MSPL. M is the index to the management class to be used. CLASM is the mgmt. class designation. PRI contains the priority designation. IXRLC is the index to rate coordinates in array RLC. All these variables are stored in /SUPC/. A percentage of the rate is eligible for full container loading (DPOFAC.3). This portion of the rate is checked against the size of container specified for the priority. If there is sufficient flow to fill a container within the specified number of days, it will be placed in dataset DPOLVL (IZNODE, ACCT, CLASS).M. This level is then checked for sufficient quantity to ship one or more containers. If the max. age is exceeded for this level, it is sent on in IFSRI(M) as an addition to the current rate. One container type (generally the smallest) may be specified for each priority class. Aggregated accounts will be split up according to AGGSF datasets. In this case, ACCT coordinate of DPOLVL with include a subaccount index to the right of the decimal point.

Assembler Inputs

Arguments

KNCONT - key to number of containers available -
0 - not constrained
1 - constrained

MSW - key to use of M-value in /SUPC/
0 - use M, CLASM in /SUPC/
.GT. 0 - use M = MSW.

MSPL - key to whether class is split into M classes

0 - not split, use NM = 1, CLASM = CLAS

1 - split, use NM, CLASM from /SUPC/

Parameter Slots

1 - load a full container and ship it.

Examples

This module can be used at a demand fill point in a continuous flow supply model.

```
CICPD (1 = FCASN (P = 0, 0, 0  $
                  1 = FCLOD (1 = CDLVL)))
```

Statistics Collected. None.

GASP Files Used. None.

Permanent Attributes Accessed

PDS Datasets:

PRIORT (node) - list of priorities of materiel to be containerized

CONTYP (node) - list of container types available

J - type of container to use for priority PRIORT (J) materiel

CONPOL (node, container type).1 - number of containers available

DPOFAC (node, class) -

1 - max hold time to obtain full container load

2 - optimum load point of container (fraction of capacity)

3 - fraction of flow which will be eligible for full container loading

CLATTR (class).1 - density of materiel in this class

CONTNR (0, container type).1 - max cube capacity of container

AGGSF (node, account) - splitting factors for aggregate account

DPOLVL (node, account, class) -

1 - level of materiel waiting in management class 1

.

.

.

M + 1 - age of materiel in management class 1 (If DPOLVL dataset does not exist, then this module creates it for M management classes)

Common/LC/:

RLC - coordinates of dataset FCRAT in use

Verb Inputs

From Calling Program

Common /LC/:

IXRLC - index to coordinates of FCRAT rate currently under consideration

From PSI. None.

Verb Outputs

To PSI

Common/CONCOM/:

ACCNT - account to load container for

CLASS - class of materiel to load container with

CONTYP - type of container to load

Common/SUPC/:

M - management class under consideration

To Calling Program. None.

Programs Called

Verbs. None.

Other. None.

Input/Output Files Used. None.

THIS PAGE LEFT INTENTIONALLY BLANK

FCLOD

General Description

This module loads a full container at a depot to be shipped directly to a port of debarkation. The information needed by the module is the account number (ACCNT), the class, and the container type (CONTYP) all stored in /CONCOM/. The materiel to be shipped is in dataset DPOLVL (IZNODE, ACCNT, CLASS).M. Priority is in PRIORI. M is the index to the management class stored in /SUPC/.

The loaded container is sent out through PS1 and can be shipped via a transportation network or immediately converted back to a flow rate.

Assembler Inputs

Arguments. None.

Parameter Slots

1 - send out loaded container (TRETC, RFLON, or CDLVL)

Examples

```
CICPD (1 = FCASN (P = 0, 0, 0  $
                  1 = FCLOD (1 = TRETC (P = 56  $
                                        1 = RTURN (P = 0),
                                            *TRANS))))
```

or

```
CICPD (1 = FCASN (P = 0, 0, 0  $
                  1 = FCLOD (1 = CDLVL)))
```

Statistics Collected

PDS Dataset CONSTS (node, container type) -

1 - number of containers shipped
2 - percent cube utilization
3 - total cube shipped
4 - total weight shipped
5 - number of consignees
6 - number of containers sent to one consignee only
8 - number of containers available

GASP Files Used. None.

Permanent Attributes Accessed

PDS Datasets:

CONTNR (0, container type).1 - max. cube for container

DPOLVL (node, account, class) -

M - cube of materiel available for shipping

NM + M - age of materiel available for shipping

CLATTR (class).1 - density of materiel to be shipped

ACCLOC (account).2 - return shipment pointer for account

Verb Inputs

From Calling Program

Common/CONCOM/:

ACCNT - account to load materiel for

CLASS - class of materiel to use

CONTYP - container type to use for loading

Common/SUPC/:

M - management class index

PDS Dataset DPOLVL (node, account, class).M - cube of materiel available for loading

From ZPLAEN

Common/ZSTACK/:

IZREF - pointer to entry in pushdown stack containing shipment in container

From PS1. None.

Verb Outputs

To ZPLAEN

Common/ZSTACK/:

IZK - weight of shipment

IZKC - identity of shipment (CLASS * CPAK1 + ACCNT)

To PS1

Common/VRBGSP/:

ATRIB - attributes of container

3 - 1000 * return shipment pointer

5 - container type

6 - priority of container

8 - stack pointer to container contents + 1000 + drop point key (2)

9 - time of container shipment

10 - cube of shipments in container

11 - weight of shipments in container

To Calling Program. None.

Programs Called

Verbs. None.

Other. ZPLAEN.

Input/Output Files Used. None.

THIS PAGE LEFT INTENTIONALLY BLANK

RCVCD

General Description

This module handles the receipt of an incoming container at a container delivery point (CDP). The first shipment is removed from the container and added to the current level of materiel for the account and class. Shipments for all classes of materiel for this account will be removed from the container. If this CDP is a drop point, then all shipments will be removed for all accounts. If this CDP serves one account, then the container will be sent on to the CDP for the next shipment in the container.

Assembler Inputs

Arguments. None.

Parameter Slots

1 - send container to the next CDP (TRETC, RFLON)

2 - dispose of the empty container

Examples

The module RCVCD is activated whenever a container is delivered to a node. It is independent of the rate/level time step execution sequence. RCVCD should be placed in a separate subnode that has been referenced by the module ADLVR to identify the container delivery point.

```
NODEA.  RLNOD (P = * NODEA, 1.),
        ADLVR (P = * NODEA, 0  $    + IDENTIFY CONTAINER
               1 = * NODEA.2),      + DELIVERY POINT
               .
               .
               .
/2/     RCVDC (1 = TRETC (P = 1$    + RECEIVE CONTAINER DELIVERY
                          1 = RTURN (P = 0), *TRANS))
```

Statistics Collected

PDS dataset CDSTAT (account, priority) -

1 - number of containers received

2 - travel time of container

3 - weight of materiel received

GASP Files Used. None.

Permanent Attributes Accessed

PDS Datasets:

CDLEVL (accounts, class).1 - level of materiel received
(If this dataset does not exist, the module creates it.).

ACCLOC (account) - location information on next account in container

Verb Inputs

From Calling Program

Common/VRBGSP/:

ATRIB - attributes of incoming container

3 - 1000. *return shipment pointer

5 - container type code

6 - priority

8 - 1000. *pointer to stack with contents of container + key to account/drop point delivery

9 - time container shipped

11 - total weight of shipments in container

From ZREMEN

Common/ZSTACK/:

IZK - weight of materiel in shipment

IZKC - identity of shipment (CLASS * CPAK1 + ACCNT)

IZREF - pointer to next shipment in container

Verb Outputs

To ZREMEN

Common/ZSTACK/:

IZREF - pointer to current top shipment in container stack

To Calling Program

Common/VRBGSP/:

ATRIB - attributes of container as input except

8 - 1000. * new pointer to stack with contents of container + key to account/drop point delivery

11 - new total weight of shipments in container

(RCVCD-2)

Programs Called

Verbs. None.

Other. ZREMEN.

Input/Output Files Used. None.

THIS PAGE LEFT INTENTIONALLY BLANK

RCVCP (IRR)

General Description

This module receives incoming shipments (that is, rates) at a CCP (consolidation and containerization point) and translates them to levels to be placed in stacks awaiting consolidation. This is the point where the continuous form is changed to a discrete form in a model.

Stacks are kept at this node for each account and priority. Any priority groupings are specified in dataset PRIORT. All classes of materiel for an account and priority are kept in the same stack. The current age of the shipment is entered and the stacks are handled on a FIFO basis, so that oldest are sent first. Statistics are collected on materiel in the stacks and the CCP.

Assembler Inputs

Arguments

IRR - index to receipt rate pointer to be used in dataset FCRAT (account, class).

Parameter Slots. None.

Examples

This module is required at a CCP node and should be executed at every rate/level time step. The argument is used to specify which receipt rate (1 - primary, 2 - secondary, etc.) is to be used for the accounts serviced by the CCP.

```
CCP.  RLNOD (P = * CCP, 81.),
      RCVCP,                    + RECEIVE SHIPMENTS AT CCP
     *RLCTR                 $
```

Statistics Collected

PDS dataset CCPSTS (node, account, priority) -

4 - cube of materiel in CCP stacks

5 - weight of materiel in CCP stacks

PDS datset CCPST (node, 0, priority) - same statistics as above for all accounts at the CCP.

GASP Files Used. None.

Permanent Attributes Accessed

PDS Datasets:

PRIORT (node) - list of priorities of materiel expected

CCPPOL (node).2 - current time at CCP, used to determine age of shipments

FCRAT (account, class) - continuous receipt rate dataset

RATATT (IXCCPR).2 - density of materiel in a receipt rate

AGGSF (nide, account) - splitting factors for an aggregate account

AGGACC (node, account, index * 100 + priority) -

1 - cube of materiel for individual account

2 - age of materiel for individual account

CDLEVL (account, class) - if CDLEVL dataset does not exist for deliveries, this module creates it.

Common/LC/:

RLC - coordinates of dataset FCRAT

Verb Inputs

From Calling Program. None.

From LRLOOP

Argument RRPTR - rate stack pointer array

Value INDEX - index to coordinates of FCRAT in array RLC

From IRSTK

Value IXCCPR - index to receipt rate stack entry for account materiel entering the CCP node (stack is stored in DTABLE).

From CPLAEN

Common/ZSTACK/:

IZREF - pointer to top entry in CCP stack for account and priority.

Verb Outputs

To LRLOOP

Argument FCRAT - name of dataset to loop on

To IRSTK

Argument KRLNOD - rate/level node index of CCP node

To CPLAEN

Common/ZSTACK/:

IZK - cube of shipment in CCP stack

IZKC - identity of shipment (CLASS * CPAK2 + AGE * CPAK3 + DNSITY)

IZREF - pointer to pushdown stack containing materiel for an account and priority

Programs Called

Verbs. None.

Other. LRLOOP, IRSTK, CPLAEN, SPLAEN.

Input/Output Files Used. None.

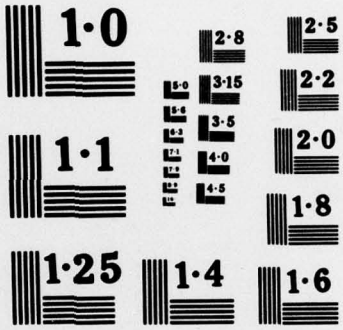
AD-A040 806 MODELS OF THE US ARMY WORLDWIDE LOGISTIC SYSTEM (MAWLOGS) VOLUME III MODU..(U) BDM CORP VIENNA VA FEB 77 BDM/W-76-211-TR-VOL-3-PT-8-9 DAAG39-76-C-0134 2/2

UNCLASSIFIED F/G 15/5 NL

END

FILMED

DTIC


1·0
1·1
1·25
1·4
1·6
1·8
2·0
2·2
2·5
2·8
3·15
3·5
4·0
4·5

THIS PAGE LEFT INTENTIONALLY BLANK

ROUCS

General Description

Determines weighting factors to be used in calculating the route from origin terminal to destination terminal for a container. The route is obtained from the Dijkstra shortest chain algorithm programmed in the routine SCHAIN. Utilizing the distance of the available links and the rate of travel for the various modes, this algorithm calculates a route which minimizes the travel time between the given terminals. Transshipment links are included in the network to properly penalize a change of modes in the routing. The priority, weight, and cube of a shipment are utilized in setting weighting factors which will block the selection of certain modes in the algorithm.

This verb should be used in conjunction with the aggregate transportation verbs RDNET, TCNTS, and TRMOP. ROUCS differs from the verb ROUTS in that it utilizes the cube of the container itself in determine mode utilization and expects the weight to be in ATRIB (II) upon entry.

Control is returned to the calling program upon completion.

Assembler Inputs

Arguments. None.

Parameter Slots. None.

Examples

This verb contains no parameter slots. It is utilized in PS1 of the verb TCNTS to calculate the initial routing of a container shipment as follows:

```
TCNTS (1 = ROUCS  $
```

Since only four link numbers of a route are saved with the temporary attributes of a shipment, the verb ROUCS may also be needed in PS2 of TRMOP. Then the route is recalculated at the terminal under the current network conditions and up to four more link numbers are saved.

Statistics Collected. None.

GASP Files Used. None.

Permanent Attributes Accessed

Common /TTERMS/:

CAPMO(I,1) - maximum weight of a shipment which can be carried on mode I

CAPMO(I,2) - maximum cube of a shipment which can be carried on mode I

PAKLNK - factor used in packing link numbers into ATRIB(7)

MODTSS - transshipment mode code

Common /TROUTE/:

BLOC(I) - the weighting factor used to block selection of mode I

HIPRI - the air priority lower limit

Verb Inputs

From Calling Program

Common /VRBGSP/:

ATRIB - attributes of shipment for which route is to be calculated, in particular

2 - source terminal *1000 + sink terminal

6 - shipment priority

10 - cube of materiel in the container

11 - weight of the materiel in the container

From WGTCUB

Arguments:

WEIGHT - the weight of the shipment

CUBE - the cube of the shipment

From SCHAIN

Common /TROUTE/:

NCHAIN - the number of links in the route calculated

LROUTE(J) - an array of the link numbers in the route calculated

Verb Outputs

To SCHAIN

Arguments:

SOURCE - source (origin) terminal of shipment

SINK - sink (destination) terminal of shipment

Common /TROUTE/:

BLOC - array of weighting factors used to block selection of modes

To Calling Program

Common /VRBGSP/:

ATRIB - attributes of shipment as input except

7 - first link in route + second link * PAKLNK + third link * $PAKLNK^2$ + fourth link * $PAKLNK^3$

Programs Called

Verbs. None.

Other. SCHAIN.

Input/Output Files Used. None.

THIS PAGE LEFT INTENTIONALLY BLANK

SNDCP (IRR)

General Description

This module is designed to convert levels of materiel at a CCP which have been consolidated into container loads back to a receipt rate flowing out of the CCP. This capability is used if the containers are not to be sent through a discrete transportation network. The logic sends container contents from the CCP through the flow logic from CDLEVL datasets for each account and class. Routine UDRR is used to update the receipt rate flowing out of this node. The receipt rate to be changed is pointed to by FCRAT (ACCT,CLASS).IRR+2.

Assembler Inputs

Arguments.

IRR - index to the return rate in FCRAT to be changed
FCRAT.(ND+3) is the first, FCRAT.(ND+4) is the second, etc.

Parameter Slots. None.

Examples

This verb must be executed in the standard rate/level simulation pass every time step.

```
CCP. RLNOD (P = * CCP, 81.),
     RCVCP (P = 1),      + RECEIVE MATERIEL AT CCP
     SNDCP (P = 1),      + SEND MATERIEL FROM CCP
     *RLCTR
```

Statistics Collected. None.

GASP Files Used. None.

Permanent Attributes Accessed

PDS Datasets:

FCRAT (account, class) - receipt rate stack pointers
CDLEVL (account, class).1 - level of materiel to be sent

Common /LC/:

RLC - coordinates of dataset FCRAT

Verb Inputs

From Calling Program. None.

From LRLOOP

Argument RRPTR - array of receipt stack pointers

Value INDEX - pointer to RLC array of FCRAT coordinates

From UDDR. None.

Verb Outputs

To LRLOOP

Argument FCRAT - name of dataset to loop on

To UDDR

Arguments -

CONRAT - rate of flow out of CCP

KRLNOD - rate/level node number for CCP

IRRPTR (IPTR) - index of receipt rate stack entry in DTABLE to update

0 - bottom switch, if KRLNOD is not found in stack, do nothing

Programs Called

Verbs. None.

Other. LRLOOP, UDDR.

Input/Output Files Used. None.

THE BDM CORPORATION

VOLUME III - MAWLOGS MODULE CATALOG

PART 9 - MODEL CHANGE MODULES

CHAPTER V
MODEL CHANGE MODULES FAMILY DESCRIPTION

A. OVERVIEW

The modules described in this part of the Module Catalog are a mixed bag of tricks, but generally relate to the capability to change a model: data values, data structure, model structure, or model logic, during the course of a model run. This capability is invaluable when simulating a dynamic system which is undergoing change, such as the transition from a peacetime status to a wartime status. By changing the model structure and data values during a model run, the analyst can observe and measure the perturbations of the system in response to the change.

The modules described here can be used with both discrete event modules and continuous flow modules in a MAWLOGS model. The modules are listed by subfamily in Table V-1. The Data Change group of modules provides the capability to change, add, or delete data throughout a model run. The Staging modules provide a capability to key different events, policies, and model structures to different stages or time periods in a model run. The DTABLE Data handling modules are an expansion of the original DTABLE capabilities and provide dynamic reallocation of array space with garbage collection. Each of these subfamilies are described in the following sections. Descriptions of the verbs in the entire family are included in Chapter VI of this document. Descriptions of the routines are included in Cahpter VII.

B. DATA CHANGE MODULES

The Data Change modules are provided so that the user has complete flexibility during a model run. Most of the data structures utilized by the MAWLOGS modules can be changed during a model run or at the restart time for a model. The general form of the change modules allows the user

TABLE V-1. MODEL CHANGE MODULES

NAME	DESCRIPTION
	DATA CHANGE MODULES
CANCEL	CANCEL A GASP FILE ENTRY
vCANCL	VERB TO CANCEL A GASP FILE ENTRY
vCHPAR	CHANGE DISTRIBUTIONS IN PARAM ARRAY DURING A MODEL RUN
vCHPDS	CHANGE PDS DATASETS DURING A MODEL RUN
vCHPDS	CHANGE PDS DATASETS DURING A MODEL RUN
vCHPMT	CHANGE PERMAT DATASETS DURING A MODEL RUN
DELDS	DELETE A SERIES OF PDS DATASETS WITH COMMON COORDINATES
MLTEL	MULTIPLY THE VALUE OF AN ELEMENT IN A PDS DATASET BY A FACTOR
RDNDS	READ A GROUP OF PDS DATASETS WITH THE SAME VALUES
	STAGING MODULES
vCHSTG	CHANGE STAGING DURING A MODEL RUN
vINSTG	INPUT VERB FOR STAGE DATASET INITIALIZATION
MSTAGE	ADVANCES STAGE NUMBERS FOR SEQUENCES WHOSE NEXT STAGE TIME IS TNOW AND SCHEDULES NEXT STAGE EVENT
vRPSTG	MODULE TO REPORT STAGING DATA
vSTGBR	BRANCH TO PROPER PARAMETER SLOT FOR CURRENT STAGE
	DTABLE DATA MODULES
DTBGC	ARRAY DTABLE GARBAGE COLLECTION IN COMMON /TBLCOM/
INDTB	ENTER ENTRIES IN A DTABLE ARRAY
RLDTB	RELEASE SPECIFIED ENTRIES IN A DTABLE ARRAY
RSETPA	RESETS POINTERS IN ARRAY PARAM TO ARRAY DTABLE AFTER GARBAGE COLLECTION
RSETRD	RESETS RATE/LEVEL DELAY BOXCAR POINTERS AFTER DTABLE GARBAGE COLLECTION
RSETRS	RESETS DTABLE POINTERS FROM LDSC, EXRAT, AND FCRAT DATASETS AFTER GARBAGE COLLECTION
	MISCELLANEOUS MODULES
CPLAEN	PLACE AN ENTRY IN AN IZHOLD STACK FOR FIFO OPERATION
IXZNOD	RETURNS INDEX OF NODE WITH A GIVEN NAME IN ARRAY ANODEL
vPRSTK	PRINT THE CONTENTS OF THE PUSHDOWN STACKS IN ARRAY IZHOLD
vRFLON	SETS UP A RETURN FLOW OF A DISCRETE SHIPMENT WITHOUT RELEASING THE RETS POINTER OR THE STACK ENTRY
vWBFIL	COMPUTES WEIBULL FUNCTION VALUES FOR A GIVEN VALUE

THE v PRECEDING A MODULE NAME INDICATES A VERB WHICH CAN BE USED IN A MODEL DESCRIPTION.

to read in groups of cards at a prespecified time to change data values or structures at particular times during the model run. The easiest way to include these modules in a model description is to include them in a subnode of the initialization node ZINIT. The ZINIT subnode can then be executed by scheduling a GASP exogenous event when a model is begun or restarted. Figure V-1 shows an example ZINIT node description which includes many of the data change capabilities. The input deck setup for multiple change decks is shown in Volume IIA, The Addendum to User's Manual. The use of alternate files for data change input streams is also possible since the file to be read is specified as an argument to the verb.

The opportunity to cancel an event that is already scheduled in the time file is provided through the verb CANCL. This verb is used mostly at model restart time to alter the event execution pattern.

C. STAGING

In order to dynamically control model logic, the concept of staging was developed. The basic idea was to set up a sequence of time periods, that could be defined in the model input data at run time, with different logical paths taken in each time period by the model. This enables links and nodes to be activated and deactivated, different policies to be utilized in wartime, and different support structures to be easily handled.

A stage sequence is defined by a PDS dataset of type STAGE which contains a sequence of times. The controlling module for the staging concept is STGBR which branches to different parameter slots based on the time values in the PDS dataset STAGE. If TNOW is less than the first time value in the STAGE dataset, parameter slot one is called. If it is less than the second time, PS2 is called, and so on. An example of the use of the verb STGBR is shown in Figure V-2. Different STAGE datasets can be used to set up different time sequences, even within the same node. The example in Figure V-2 shows parameter slot 2 (PS2) of VERBA filled with three alternative verbs to be executed in the different stages defined by

```
ZINIT.  ISTAT, IPDS, TINIT, . . .

/2/       FILE (P = 1 $ 1 = RTURN (P = 0), *ZINIT.4)     + CANCEL AN EVENT

/3/       CHPDS (P = 5), CHPAR (P = 5)    + CHANGE PDS AND PARAMS

/4/       CANCL (P = 1)       + CANCEL AN EVENT

/5/       CHPAR          + CHANGE PARAMS
```

Figure V-1. Example Node ZINIT With Data Change Capability

```
VERBA (1 = VERBB $
       2 = STGBR (P = 13 $
                  1 = VERBC $ + 1st TIME PERIOD LOGIC
                  2 = VERBD $ + 2nd TIME PERIOD LOGIC
                  3 = VERBE )), + 3rd TIME PERIOD LOGIC

STGBR (P = 4 $
       1 = DELAY (P = 0), *NODEB $ + 1st TIME PERIOD SUPPORT NODE
       2 = DELAY (P = 0), *NODEC )$ + 2nd TIME PERIOD SUPPORT NODE
```

Figure V-2. Example of Node With Staging

STAGE dataset 13. It also shows two different support nodes defined during two stages defined by STAGE dataset 4. The staging capability permits great flexibility in describing a model while leaving the specification of the stage change times until model execution time.

To make the staging operation as efficient as possible, the data structure for staging and the MSTAGE routine were defined to eliminate searching of the STAGE datasets for each execution of the verb STGBR. The common block /ZSTAGE/ is defined in Table V-2. The array STHSH is set up in the model to include every time which indicates a change in stage for at least one stage sequence. An MSTAGE event is then scheduled for each of these times to define the array IZSTAG properly for the next time period. The array IZSTAG contains the number of the parameter slot to be called at the current time for each STAGE dataset sequence. As an example, consider the STAGE datasets:

STAGE (1) = (1.0, 3.0, 4.1)

STAGE (2) = (2.0, 3.0, 5.0)

The array STHSH would contain 5 entries:

STHSH = (1.0, 2.0, 3.0, 4.1, 5.0)

From time 0.0 to 1.0, the following values would be set:

ISTAG = 1

IZSTAG = (1, 1)

and PS1 of STGBR would be called for both STAGE sequences. At time 1.0, an MSTAGE event would occur and the following values would be set:

ISTAG = 2

IZSTAG = (2,1)

and any references to STGBR for sequence 1 would cause PS2 to be executed. As one final example, at time 4.5, the values would be:

ISTAG = 5

IZSTAG = (3, 2).

Thus, any execution of STGBR for the first sequence would activate the verbs in PS3 and for the second STAGE sequence would activate the verbs in PS2. This procedure minimizes the searching of the STAGE datasets to recognize the stage change times.

TABLE V-2. COMMON BLOCK /ZSTAGE/

NAME	DESCRIPTION
IZSTAG (*NSTAG)	THE STATUS OF EACH STAGE SEQUENCE, THAT IS, THE NUMBER OF THE PARAMETER SLOT OF STGBR TO BE USED AT THE CURRENT TIME
NSTAG	THE MAXIMUM NUMBER OF STAGE SEQUENCES IN THE MODEL (I.E., THE NUMBER OF STAGE DATASETS)
STHSH (*LSTAG)	THE COMPLETE SEQUENCE OF TIMES SPECIFIED IN ALL STAGE DATASETS IN THE MODEL
LSTAG	THE MAXIMUM LENGTH OF THE STHSH ARRAY, THAT IS, THE MAXIMUM NUMBER OF NON-DUPLICATE TIMES IN THE STAGE DATASETS
ISTAG	INDEX TO THE LOCATION OF THE NEXT TIME SPECIFIED IN THE STHSH ARRAY CLOSEST TO TNOW

D. DTABLE DATA HANDLING

The original use of DTABLE did not include the storage of rate/level linkage stacks. This use required a more dynamic handling of storage allocation for table entries in the array DTABLE. The group of modules which were designed are compatible with the original uses of DTABLE and provide allocation and reallocation through garbage collection of the storage areas in the array DTABLE. Since a number of modules depend on pointers into the array DTABLE, the movement of tables during the garbage collection requires resetting these pointers. To facilitate this a mapping is created during the garbage collection process. The common block /DTBMAP/ holds the data for this mapping and is described in Table V-3. The routines RSETPA, RSETRD, and RSETRS utilize this mapping to reset the proper pointers.

TABLE V-3. COMMON BLOCK /DTBMAP/

NAME	DESCRIPTION
DTMAP (*NDTMAP, 2)	DTABLE GARBAGE COLLECTION MAP:
DTMAP (•, 1)	PREVIOUS LOCATION OF A TABLE IN DTABLE THAT HAS BEEN MOVED
DTMAP (•, 2)	NUMBER OF LOCATIONS THAT THE TABLE HAS BEEN MOVED UP IN DTABLE FROM ITS ORIGINAL LOCATION
IDTMAP	INDEX OF LAST ENTRY IN ARRAY DTMAP
IDTMAX	MAXIMUM NUMBER OF ENTRIES POSSIBLE IN DTMAP ARRAY (SET TO *NDTMAP)

CHAPTER VI

MODEL CHANGE VERBS

THIS PAGE LEFT INTENTIONALLY BLANK

CANCL (NFILE)

General Description

This verb cancels an event in the GASP file NFILE which has characteristics specified in array ATRIB. It will match all attributes in ATRIB against file entries. If ATRIB contains code -99., then any value will be accepted as a match. ATRIB(1) is the time of cancellation for the event. ATRIB(2) through ATRIB(IATT) are matched against attributes 1 through IATT-1 of the entries in the file specified.

Assembler Inputs

Arguments.

NFILE - GASP file in which event is to be cancelled.

Parameter Slots. None.

Examples

This capability is generally used to cancel events in the time file that are waiting to occur when a model is restarted. The cancellation cards are read in through the FILE verb and sent to the verb CANCL.

```
ZINIT.
    /2/  FILE (P = 1 $ 1 = RTURN (P=0), *ZINIT.3)
    /3/  CANCL (P = 1)  + CANCEL AN EVENT IN TIME FILE
```

Statistics Collected

None.

GASP Files Used

NFILE - argument to be specified

Permanent Attributes Accessed

In COMMON/VRBGSP/

ATRIB - file entry attributes

IATT - number of attributes to be tested

In COMMON/MAWVRB/

HOLD - file entry attributes

Verb Inputs

From Calling Program.

Argument

NFILE - GASP file name from which event is to be cancelled

In COMMON/VRBGSP/

ATTRIB - GASP file entry attributes -

1 - time of cancellation

2 - values to be matched against SETS (1) -

. SETS (IATT-1) of event to be cancelled

.

.

IATT

IATT - number of attributes

Verb Outputs

To CANCEL.

Arguments

HOLD (1) - HOLD (IATT) - attributes of entry to be cancelled

NFILE - number of file that entry is in

KEY - key to show whether event was found and cancelled

(0-no, 1-yes)

To Calling Program. None.

Programs Called

Verbs. None.

Other. CANCEL.

Input/Output Files Used

None.

CHPAR (INFILE)

General Description

The probability distributions which are stored in arrays PARAM and DTABLE may be changed or be newly defined at any time during a model run. Either the parameter values for the distribution, the type of distribution or both may be changed with the CHPAR module. A deck of CHPAR cards will be read whenever a subnode of node ZINIT containing the CHPAR verb is executed by an exogenous event card in the DATAN deck. The actual parameter changes will occur at the time specified on the CHPAR cards. The format of CHPAR cards are shown in the attached tables.

Assembler Inputs

Arguments.

INFILE - file name containing CHPAR card images.

Parameter Slots. None.

Examples

The CHPAR module is generally placed in a subnode of ZINIT so it can be scheduled exogenously:

```
ZINIT.
     /2/  CHPAR (P=5)     +CHANGE PARAM DATA
```

The card images can be read from any file specified, thereby not requiring a fixed order in the main input data stream.

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

Common /VRBGSP/:

PARAM - array containing distribution parameters

Common /GSPCOM/:

NPRMS - max number of parameters possible in array PARAM

Common /TBLCOM/:

DTABLE - array containing empirical (tabular) distribution values

Verb Inputs

From Calling Program. None.

From File INFILE.

New distribution values as specified on CHPAR cards with the formats specified in attached table.

From RLDTB. None.

From INDTB.

Argument IDT - index to location in DTABLE of new table.

Verb Outputs

To RLDTB.

Arguments -

First - location of table in DTABLE to be released

Second - length of table to be released

To INDTB.

Arguments -

First - array of new table entries

Second - number of entries in new table.

Programs Called

Verbs. None.

Other. INDTB, INEROR, RLDTB, RPTRV, ZERRPT.

Input/Output Files Used

INFILE - file containing change values to be read in

6 - system file for printed reports.

TABLE. CHPAR1 AND CHPAR2 DATA CARD FORMATS

CARD TYPE 1 - CHANGE SET IDENTIFICATION CARD, ONE REQUIRED

COLUMN	FORMAT	DESCRIPTION
1 - 6	A6	ENTER CHPAR1
7	1X	BLANK
8 - 25	3A6	NAME OF CHANGE SET (E.G., TIME, PURPOSE,...)
26 - 30	I5	NON-FATAL SWITCH (0 - ERRORS ARE FATAL, 1 - ERRORS ARE FLAGGED ONLY)
31 - 80	50X	BLANK

CARD TYPE 2 - DISTRIBUTION PARAMETER CARDS - ONE FOR EACH DIST. TO BE CHANGED

COLUMN	FORMAT	DISTRIBUTION
1 - 6	A6	ENTER CHPAR2
7	1X	BLANK
8 - 12	A5	DISTRIBUTION TYPE *
13	1X	BLANK
14 - 16	I3	DISTRIBUTION INDEX NUMBER **
17	1X	BLANK
18 - 25	F8.0	FIRST PARAMETER
26 - 33	F8.0	SECOND PARAMETER
34 - 41	F8.0	THIRD PARAMETER
42 - 49	F8.0	FOURTH PARAMETER
50 - 80	31X	BLANK

* - SEE FOLLOWING TABLE FOR ALLOWABLE TYPES AND PARAMETERS
** - THE NUMBER USED IN THE MODEL DESCRIPTION AND/OR INPUT

TABLE. RANDOM VARIABLE TYPES AND PARAMETERS

DISTRIBUTION	NAME FOR INPUT	FIRST PARAM.	SECOND PARAM.	THIRD PARAM.	FOURTH PARAM.
NORMAL	NORML	MEAN	MINIMUM	MAXIMUM	STANDARD DEVIATION
LOGNORMAL	LGNOR	MEAN	MINIMUM	MAXIMUM	STD. DEV.
ERLANG	ERLNG	MEAN	MINIMUM	MAXIMUM	ERLNG PARAMETER
POISSON	POSSN	MEAN	MINIMUM	MAXIMUM	BLANK
GEOMETRIC	GEOMT	MEAN	MINIMUM	MAXIMUM	BLANK
CONSTANT	CONST	VALUE	BLANK	BLANK	BLANK
EMPIRICAL DATA **	TABLE	BLANK	NUMBER OF POINTS IN DISTRIBUTION	BLANK	TYPE INDICATOR *
END-OF-CHECK	**END				

* - ENTER 0 IF THE DISTRIBUTION IS CONTINUOUS, ENTER 1 IF THE DISTRIBUTION IS DISCRETE.

** - WHEN AN EMPIRICAL DISTRIBUTION IS USED, THE DIST. PARAMETER CARD MUST BE IMMEDIATELY FOLLOWED BY A SERIES OF TYPE 3 CARDS DEFINING THE POINTS IN THE DISTRIBUTION.

TABLE. CHPAR3 DATA CARD FORMAT

CARD TYPE 3 - DISTRIBUTIONS DESCRIBED IN TABULAR FORM

COLUMN	FORMAT	DESCRIPTION
1 - 6	A6	ENTER CHPAR3
7 - 8	2X	BLANK
9 - 14	F6.0	MINIMUM VALUE OF DISTRIBUTION
15 - 18	F4.4	PROBABILITY THAT MINIMUM VALUE WILL OCCUR
19 - 24	F6.0	SECOND POINT IN DISTRIBUTION
25 - 28	F4.4	PROBABILITY THAT THE RANDOM VARIABLE WILL NOT EXCEED SECOND VALUE (CUMULATIVE PROB.)
29 - 34	F6.0	THIRD POINT
35 - 38	F4.4	CUMULATIVE PROBABILITY OF THIRD POINT
39 - 44	F6.0	FOURTH POINT
45 - 48	F4.4	CUMULATIVE PROBABILITY OF FOURTH POINT
49 - 54	F6.0	.
55 - 58	F4.4	.
59 - 64	F6.0	.
65 - 68	F4.4	.
69 - 74	F6.0	SEVENTH POINT
75 - 78	F4.4	CUMULATIVE PROBABILITY OF SEVENTH POINT
79 - 80	2X	BLANK

CONTINUATION CARD

1 - 6	A6	ENTER CHPAR3
7 - 8	2X	BLANK
9 - 14	F6.0	EIGHTH POINT
15 - 18	F4.4	CUMULATIVE PROBABILITY OF EIGHTH POINT
.	.	.
.	.	.
.	.	.

CONTINUE CODING DISTRIBUTION POINTS, SEVEN TO A CARD, USING AS MANY CARDS AS REQUIRED. THE FIRST POINT IN THE DISTRIBUTION WHICH HAS A CUM. PROB. OF 1.0 IS CONSIDERED AS THE END OF TABLE.

THIS PAGE LEFT INTENTIONALLY BLANK

CHPDS (INFILE)

General Description

This module reads cards to change PDS datasets from file INFILE. If an error occurs the run will be terminated unless the nonfatal switch is .GT. 0 on the CHPDSI card. The changes can be scheduled at any time during a model run. The type of changes that can be performed are as follows:

CHGEL - change the value of an element of a dataset

SETEL - set a new value for an element

SETSET - set new values for all elements of a dataset

ADDSET - add a new dataset to the model

RELSET - release a dataset from the model

MLTEL - multiply the value of an element by a factor.

The CHPDS cards are read from a specified file so that they are not necessarily a part of the main input stream. The format for input cards is shown in the accompanying table.

Assembler Inputs

Arguments.

INFILE - number of file that card images will be read from.

Parameter Slots. None.

Examples

The CHPDS verb can be included in a subnode of ZINIT so that execution of it can be scheduled exogenously with a -8 event.

```
ZINIT.
    /2/  CHPDS (P=5)     + READ CHPDS FROM CARD READER
    /3/  CHPAR (P=4), CHPDS (P=4)    + READ CHANGES FROM FILE 4
```

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

Any PDS dataset referenced on a CHPDS card.

Verb Inputs

From Calling Program.

None.

Verb Outputs

To PDS Datasets.

Changes to existing datasets and additions and deletions of datasets as specified on the input cards.

To Calling Program. None.

Programs Called

Verbs. None.

Other. CLEAR, CLER1, ADDSET, CHGEL, GETSET, MLTEL, RELSET, SETEL, SETSET, ZERRPT, CHRLST, LRNIX, RLINT, RLSSET, DELDS.

Input/Output Files Used

INFILE - file from which card images are read.

TABLE. CHPDS DATA CARD FORMATS

CHPDS1 CARD - ONE FOR EACH CHPDS DECK		
CC 1-CC 6	A6	ENTER CHPDS1
CC 7-CC18	2A6	IDENTIFIER FOR CHANGE SET
CC19-CC20	I2	NON-FATAL SWITCH (0-ERRORS ARE FATAL, 1-INPUT ERRORS ARE FLAGGED ONLY)
CHPDS2 CARD - ONE FOR EACH DATASET TO BE CHANGED		
CC 1-CC 6	A6	ENTER CHPRM2
CC 7-CC12	A6	TYPE OF CHANGE - CHGEL, SETEL, SETSET, ADDSET, RELSET OR MLTEL (***END FOR LAST CARD IN DECK)
CC13-CC18	A6	DATASET NAME
CC19-CC20	I2	NUMBER OF THE ATTRIBUTE TO BE CHANGED OR NUMBER OF ATTRIBUTES IN THE DATASET FOR A "SET" OPERATION
CC21-CC28	F8.0	VALUE TO BE USED IN ELEMENT CHANGE OPERATION
CC29-CC30	I2	RATE/LEVEL STATISTIC CHANGE CODE FOR ELEMENT OPERATION (-1 - TURN STATISTIC OFF 0 - NO CHANGE POS - CHANGE FREQUENCY AND TYPE TO NEW VALUE)
CC31-CC40	F10.3	FIRST COORDINATE OF DATASET
CC41-CC50	F10.3	SECOND COORDINATE OF DATASET
.	.	
.	.	
.	.	
CC71-CC80	F10.3	FIFTH COORDINATE OF DATASET
CHPDS3 CARD - ONE OR MORE REQUIRED FOR SETSET OR ADDSET OPERATION, FOLLOWING CHPDS2 CARD		
CC 1-CC 6	A6	ENTER CHPDS3
CC 7-CC12	A6	TYPE OF CHANGE (SAME AS ON CHPDS2 CARD)
CC13-CC20	8X	BLANK
CC21-CC28	F8.0	VALUE OF FIRST, SEVENTH, OR THIRTEENTH ELEMENT
CC29-CC30	I2	RATE LEVEL STATISTIC CHANGE CODE
.	.	
.	.	
.	.	
CC71-CC78	F8.0	VALUE OF SIXTH, TWELFTH, OR EIGHTEENTH ELEMENT
CC79-CC80	I2	RATE LEVEL STATISTIC CHANGE CODE

THE DECK FOR A SINGLE EXECUTION OF CHPDS CONSISTS OF ONE CHPDS1 CARD, A SERIES OF CHPDS2 CARDS, AND A FINAL CHPDS2 CARD WITH ***END IN CC7-CC12. IF A CHPDS2 CARD IS FOR A SETSET OR ADDSET, THEN IT MUST BE IMMEDIATELY FOLLOWED BY ONE OR MORE CHPDS3 CARDS WITH THE VALUES OF THE ELEMENTS IN THE SET.

THIS PAGE LEFT INTENTIONALLY BLANK

CHPMT (INFILE)

General Description

Any value in a PERMAT permanent attribute set that was read in on an INITP card can be changed during the model run utilizing the CHPRM module. This change PERMAT module can also be used to add or delete entire data sets for any of the nodes in the model. This feature can be used, for example, to add a supply class to the inventory for a supply nodes. The format of CHPRM cards is shown in the accompanying table. Each CHPRM deck consists of a CHRPM1 card that labels it, followed by a sequence of CHPRM2 cards that specify the data set to be changed by node name and data set name. If more than 4 elements of a data set are to be changed, one or more continuation cards of type CHPRM3 must follow a CHPRM2 card. The last CHPRM2 card in the deck must have ***END in CC7-CC12. The type of change to be performed is specified on each CHPRM2 card with the following codes:

CHGEL - change one element (add the given value to the existing value)

SETEL - set one element (set the value to the given value)

SETSET - set the values of a set of attributes

ADDSET - add the set of attributes to the model

RELSET - release a set of attributes from the model

MLTEL - multiply the value of one element by the given value.

A deck of CHPRM cards will be read whenever the CHPRM module is executed. The actual changes will occur at that time.

Assembler Inputs

Arguments. INFILE - file that card images are to be read from.

Parameter Slots. None.

Examples

The CHPRM module can be included in a subnode of the ZINIT node so that execution can be scheduled by an exogenous event of type -8.

ZINIT.

/2/ CHPMT (INFILE)

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

PERMAT datasets specified on change cards.

Verb Inputs

From Calling Program. None.

From RESID.

Function value - PERMAT resource identifier to be used as second coordinate.

Verb Outputs

To PERMAT Datasets. New element value for an existing dataset or addition or deletion of datasets.

To Calling Program. None.

To RESID.

Arguments

RSFCN - PERMAT resource function name

NUMRES - resource number

FCNTYP - function type (MNOD, SNOD, ...)

Programs Called

Verbs. None.

Other. PERMDS, RESID, ADDSET, CHGEL, GETSET, MLTEL, RELSET, SETEL, SETSET, CLEAR.

Input/Output Files Used

None.

TABLE. CHPRM DATA CARD FORMATS

CHPRM1 CARD -		
CC 1-CC 6	A6	ENTER CHPRM1
CC 7-C12	A6	OPTIONAL NAME FOR THIS CHANGE SET
CC13-C14	I2	NON-FATAL SWITCH (0-ERRORS ARE FATAL, 1-INPUT ERRORS ARE FLAGGED ONLY)
CHPRM2 CARD -		
CC 1-CC 6	A6	ENTER CHPRM2
CC 7-CC12	A6	TYPE OF CHANGE - CHGEL, SETEL, SETSET, ADDSET, RELSET OR MLTEL. (***END FOR LAST CARD)
CC13-CC17	A5	NAME OF NODE DATASET IS ASSOCIATED WITH OR **ALL
CC18-CC23	A6	PERMAT RESOURCE FUNCTION NAME (E.G., SUPAR, SICOM)
CC24-CC33	I10	RESOURCE IDENTIFIER (E.G., ITEM NUMBER)
CC34-CC37	A4	FUNCTION TYPE (E.G., SNOD, SNOI, MNOD)
CC38-CC40	I3	ATTRIBUTE NUMBER OF NUMBER OF ATTRIBUTES
CC41-CC50	F10.0	SINGLE VALUE TO BE USED OR FIRST ATTRIBUTE
(FOLLOWING ONLY USED FOR ADDSET AND SETSET)		
CC51-CC60	F10.0	SECOND ATTRIBUTE
CC61-CC70	F10.0	THIRD ATTRIBUTE
CC71-CC80	F10.0	FOURTH ATTRIBUTE
CHPRM3 CARD - ONLY USED FOR ADDSET OR SETSET WITH MORE THAN 4, 11, OR 18 ATTRIBUTES.		
CC 1-CC 6	A6	ENTER CHPRM3
CC 7-CC10	4X	NOT USED
CC11-CC20	F10.0	FIFTH, TWELFTH, OR NINETEENTH ATTRIBUTE
CC21-CC30	F10.0	SIXTH, THIRTEENTH, OR TWENTIETH ATTRIBUTE
CC31-CC40	F10.0	SEVENTH OR FOURTEENTH ATTRIBUTE
CC41-CC50	F10.0	EIGHTH OR FIFTEENTH ATTRIBUTE
CC51-CC60	F10.0	NINTH OR SIXTEENTH ATTRIBUTE
CC61-CC70	F10.0	TENTH OR SEVENTEENTH ATTRIBUTE
CC71-CC80	F10.0	ELEVENTH OR EIGHTEENTH ATTRIBUTE

LAST CARD IN DECK IS A CHPRM2 CARD WITH ***END IN CC 7-CC12.
FOR A VACUOUS DECK, THE ***END CARD MAY BE A CHPRM1 CARD.

THIS PAGE LEFT INTENTIONALLY BLANK

CHSTG (IZINIT)

General Description

Assumes new stage datasets have been read in and cancels next stage event in the time file. The stage sequencing arrays are redefined from the datasets for the current time. The new next stage event is scheduled in the time file. The module permits redefining any stage sequence times in the model during the model run or at restart time. The STAGE dataset values should be changed by CHPDS before CHSTG is executed.

Assembler Inputs

Arguments.

IZINIT - subnode number of node ZINIT which may be executed when an MSTAGE event occurs. This overrides the original subnode specification from INSTG.

Parameter Slots. None.

Examples

This model can be placed in a subnode of ZINIT so that it can be scheduled for execution by an exogenous event.

```
ZINIT. ...
     /4/  CHSTG (P=7)     + CHANGE STAGING
     /7/  CHPDS (P=5), CHPAR (P=5)    + DATA CHANGES
```

Statistics Collected

None.

GASP Files Used

File 1 - simulation time file

Permanent Attributes Accessed

COMMON /VRBGSP/:

IATT - number of attributes in ATRIB

TNOW - current simulation time

COMMON /ZSTAGE/ - all elements

COMMON /DSTOR/

DSPOOL - array where PDS datasets are stored, STAGE datasets referenced directly

Verb Inputs

From Calling Program. None.

From CANCEL.

Argument KEY to verify cancellation (0-no, 1-yes)

From IDSLON.

Arguments -

INDEX - location of STAGE dataset in DSPOOL array

LEN - length of STAGE dataset

From SORTF.

Argument STHSH - sorted sequence of stage change times

Verb Outputs

To FILEM.

Arguments.

1 - number of file to use (TIME file)

HOLD - attributes of stage sequence

1 - next stage event time

2 - -9.

3 - subnode of ZINIT for stage changes

To CANCEL.

Arguments

HOLD - attributes of old MSTAGE event in time file

To IDSLON.

Argument STAGE - name of dataset to loop on

To HASHF.

Arguments - identification of new entry for STHSH array (used to avoid redundant times)

To SORTF.

Arguments

STHSH - array to be sorted

LSTAG - max number of entries in array

Programs Called

Verbs. RPSTG

Other. CLEAR, CANCEL, IZSTOR, IDSLON, HASHF, SORTF, FILEM.

Input/Output Files Used

None.

THIS PAGE LEFT INTENTIONALLY BLANK

INSTG (IZINIT)

General Description

Initializes stage data structure. Sorts all STAGE sequence times and adds initial MSTAGE event to the time file. This verb should be executed after the STAGE datasets have been defined through a verb such as IPDS.

Assembler Inputs

Arguments.

IZINIT - subnode indicator of node ZINIT for data changes that can be executed at stage change times

Parameter Slots. None.

Examples

The verb INSTG should be executed one time, following the definition of STAGE datasets.

```
ZINIT.  ISTAT, IPDS, INPAR, INSTG (P=3), RPSTG
/3/  CHPDS (P=5), CHPAR (P=5)    + CHANGE DATA AT
                                 + STAGE CHANGES
```

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

COMMON /ZSTAGE/: all elements

COMMON /PDSRCH/:

HLOCK - logical flag

COMMON /DSTOR/:

DSPOOL - array where PDS datasets are stored, STAGE datasets referenced directly

Verb Inputs

From Calling Program. None.

From IDSLON.

Arguments -

I - location of STAGE dataset in DSPOOL
0 when all datasets have been found
-1 if there are no datasets

LEN - number of elements in dataset

From HASHF.

Arguments -

IADD - biased location of DSPOOL (I+J-1), or -1 if datum could not be found in search mode, or -3 if table was found to be full in store mode

STHSH - DSPOOL (I+J-1) stored at IADD if a store attempt did not find the data already present and the table was not full.

Verb Outputs

To Permanent Attributes

COMMON /ZSTAGE/:

IZSTAG - stage sequence status array, IZSTAG(6)=3 indicates that PS3 is to be activated by STGBR for sequence 6 at the current time

STHSH - array of event times for all staging sequences

COMMON /PDRSCH/:

HLOCK - logical flag

To MSTAGE

COMMON /VRBGSP/:

ATRIB(3) - subnode of ZINIT to be executed by MSTAGE event

Programs Called

Verbs. None.

Other. IZSTOR, IDSLON, HASHF, SORTF, MSTAGE.

Input/Output Files Used

NPRNT - FORTRAN logical file for printed reports.

PRSTK

General Description

This verb calls the routine STKPR which writes on file 6 the current contents of the stack variables IZFREE, IZREF, IZK, and IZKC (unpacked), and the contents of IZHOLD in table form up to IZHOLD (j, IMAX) or to the end of the array.

Assembler Inputs

Arguments. None.

Parameter Slots. None.

Examples

This verb was created so that a pushdown stack dump of IZHOLD could be obtained on command. As an example, this could be included in the error subnode of ZRPRT or in a separate subnode of ZINIT that could be scheduled exogenously.

```
ZRPRT. ...
     /2/ ..., PRSTK, ...
```

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

In COMMON/VRBGSP/

NZHOLD - maximum number of three word entries the pushdown may contain.

Verb Inputs

From Calling Program. None.

Verb Outputs

To STKPR.

Arguments:

NZHOLD - maximum number of entries in IZHOLD array (PRSTK-1)

0 - fatal switch, 0 indicates non-fatal dump, return to PRSTK.

To Calling Program. None.

Programs Called

Verbs. None.

Other. STKPR.

Input/Output Files Used

None.

RFLON

General Description

Creates a return flow event, i.e., an event constituting a "response" to a logistic "demand," and schedules the occurrence of the event through the time file. The destination and delay time of the return flow must have been recorded as attributes of the demand at the time the demand was generated. Verb RNOD is available for this purpose. If no return destination and delay time have been specified for the "demand" event, a message is printed and control is given to the time file; no return flow event is generated. Otherwise, control is returned to the calling program. RFLON differs from RFLOW only by the handling of the RETS pointer in ATTRIB(3). (RFLON = RFLOW with no release of the stack entry).

RFLON - utilizes RETS pointer to read stack entry only. Does not change the value in ATRIB(3).

RFLOW - utilizes RETS pointer to remove stack entry, uses the data and discards it. The pointer to the next entry in the stack is placed in ATRIB(3).

RFLON sets up a return flow event using data stored by RNOD/RNOD1, or ADLVR, and files the event in the time file, in particular, sets attributes 1, 3, and IATT+1 of the return event as follows -

ATRIB(1) - TNOW + time obtained through DISTR(IZSTOC) (time of occurrence)

ATRIB(3) - IZSTK*1000., where IZSTK is pointer to next entry in return stack

ATRIB(IATT+1) - pointer to entry in logic stacks containing data needed to access the logic for processing the return event when it occurs.

RFLON sets the stack entry pointed to from ATRIB(IATT+1) as follows -

IZK - ILINK, the address of a CALL LINK(1) in RFLON, which call is the implementation of PS1 of ADLVR or RNOD/RNOD1

IZKC - IZNODE + IZVERB*IP12, where IZNODE and IZVERB have the values originally saved by RNOD/RNODI, i.e., the values that cause a CALL LINK(1) to execute the content of PS1 of RNOD/RNODI.

RFLO assumes that, when called, ATRIB(3) contains to the left of the rightmost three digits a value IZSTK which is the index of an entry in the stacks containing

IZKC = IZSTOC*IP24 + IZVERB*IP12 + IZNODE

IZK = -IZSNOD

Where IZSTOC represents a delay time to be accessed through DISTR, and IZVERB, IZNODE, and IZSNOD have the values saved by RNOD/RNODI.

Assembler Inputs

Arguments. None.

Parameter Slots. None.

Examples

The verb RFLON is used to set up a return flow shipment but not destroy the stack entry in IZHOLD that stores the return address for the shipment. This verb has been used with the containerization modules that set a return address once with the verb ADLVR.

```
ACCTA. RLNOD (P = *ACCTA, 13.),
       ADLVR (P = *ACCTA, 0 $    + IDENTIFY CONTAINER
                                 + DELIVERY POINT
              1 = *ACCTA.2), ...
/2/ RCVCD (1 - TRETC (P = 13 $  + RECEIVE A CONTAINER DELIVERY
                      1 = RTURN (P = 0), *TRANS)
TRANS.
/2/ TRMOP (1 = DELAY (P = 0),    + TERMINAL OPERATIONS
           DELVS (1 = RFLON) $ + DELIVERY CONTAINER
```

Statistics Collected

None.

GASP Files Used

Time file.

Permanent Attributes Accessed

In /VRBGSP/ - through a call to verb DISTR:

PARAM(IZSTOC,J), J = 1,2,3,4 - the parameters of the random variable with index ISTOC. This random variable is assumed to represent the return flow delay time. If the random variable is in tabular form, its portion of DTABLE in /TBLCOM/ will be accessed.

Verb Inputs

From Calling Program.

In /VRBGSP/ - ATRIB, attributes of "demand" event for which a return flow event is to be generated, including 3 - IZSTK X 1000, where IZSTK is the return flow pointer, i.e., the index in IZHOLD of the entry containing the return flow destination and delay time.

From Verb DISTR.

In /ZMAWSY/ -

ZSWT - the return flow delay time.

From ZPLAEN.

In /ZSTACK/ -

IZREF - pointer to new stack entry created

From FILEM. None.

Verb Outputs

To Verb DISTR.

Argument IZSTOCK - index of the random variable representing the delay time distribution from which a value is to be chosen for the return flow event

To ZPLAEN.

In /ZSTACK/ - a stack entry representing the logic flow path to which program control will be transferred when the return flow event occurs

IZREF - 0, to start a new logic flow stack

IZK - IZLINK, the core address of the CALL LINK(1) in RFLON by which PS1 of the RNOD or ADLVR represented by IZVERB will be executed

IZKC - IZVERB X IP12) + IZNODE, where IZVERB identifies the RNOD reference in which the return destination was specified, the IZNODE is the number of the node in which the RNOD reference appeared

To Time File through FILEM.

In /VRBGSP/ - ATRIB, attributes of return flow event, including attributes of "demand" as input from the calling program, except for:

1 - TNOW + ZSWT, where ZSWT is the return flow delay time obtained through vDISTR

In /ZMAWSY/ -

IZRET - index in stacks array of logic flow entry filed by ZPLAEN as described immediately above; FILEM will store this index in ATRIB (IATT + 1) in the usual manner for time file entries

To Calling Program.

In /VRBGSP/ - ATRIB attributes of return flow event

To Permanent Attributes. None.

Programs Called

Verbs. DISTR

Other. FILEM, ZFADE, ZPLAEN.

Input/Output Files Used

NPRNT - print file.

RPSTG

General Description

Reports the status of the staging sequence. It reports the time sequence of stage events and the status of each stage sequence for the current time. The figure shows a sample report printed by this routine.

Assembler Inputs

Arguments. None.

Parameter Slots. None.

Examples

This verb can be placed anywhere in a model description where the status of the staging sequences is desired, but most often will be used in a subnode of the reports node ZRPRT.

ZRPRT. ..., RPSTG, ...

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

Common /ZSTAGE/ - all data

Common /VRBGSP/:

TNOW - 'current' simulation time

Verb Inputs

None.

Verb Outputs

None.

Programs Called

Verbs. None.

Other. None.

Input/Output Files Used

6 - system print file.

THIS PAGE LEFT INTENTIONALLY BLANK

STGBR (IXX)

General Description

This is the controlling module for the staging concept. It branches to different parameter slots based on the time values in the PDS dataset STAGE (IXX). If TNOW is less than the first time value in the STAGE dataset, parameter slot one is called. If it is less than the second time, PSZ is called, and so on. If time is greater than last time value in dataset, parameter slot n+1 is called where n equals the number of time entries in dataset.

Assembler Inputs

Arguments.

IXX - stage sequence number

Parameter Slots.

Call PS1 if TNOW < T1

Call PS2 if $T1 \leq TNOW < T2$

. . .

. . .

. . .

Call PS20 if $T19 \leq TNOW$

Examples

Staging can be used to branch around a node, effectively turning it off for a time period.

```
NODEA. STGBR (1 = *NEXTN $ + BRANCH TO NEXT NODE
              Z = *NODEA.2) + EXECUTE NODEA
```

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

Common /ZSTAGE/:

IZSTAG - stage sequence array to determine PS for current time

NSTAG - number of elements in IZSTAG

Verb Inputs

None.

Verb Outputs

None.

Programs Called

Verbs. None.

Other. ZERRPT.

Input/Output Files Used

None.

WBFIL

General Description

Computes Weibull function F(x) given X and Weibull parameters α, β, γ. A simple Weibull cumulative distribution function is defined as,

$$F(x) = 1-e^{-[(x-\gamma)^{\beta}]/\alpha}$$

for $x \geq \gamma$; α, β positive

Assembler Inputs

Arguments. None.

Parameter Slots. None.

Statistics Collected

None

GASP Files Used

None

Permanent Attributes Accessed

COMMON /ZMAWSY/:

IZSWT - location of Weibull parameters in PARAM (IZSWT,*)

ZSWT - x value used to compute Weibull F(x)

COMMON /VRBGSP/:

PARAM - distribution parameters

(IP,1) - scale parameter, α

(IP,2) - shape parameter, β

(IP,3) - location parameter, γ

Verb Inputs

From Calling Program.

COMMON /ZMAWSY/:

IZSWT - location of Weibull parameters in PARAM (IZSWT,*)

ZSWT - x value used to compute Weibull F(x)

Verb Outputs

COMMON /ZMAWSY/:

ZSWT = Weibull function value

Programs Called

Verbs. None.

Other. None.

Input/Output Files Used

None.

CHAPTER VII

MODEL CHANGE ROUTINES

THIS PAGE LEFT INTENTIONALLY BLANK

CANCEL (DUMAT, NFILE, KEY)

General Description

This routine is used to cancel file entries which are identified by specified attributes. A flag is set when an entry is cancelled and control is returned to the calling program. The attributes of the cancelled event are printed out with a message.

Arguments

DUMAT - attributes of entry in NFILE to be found and cancelled. If DUMAT contains code -99. in a position, then any value will be accepted as a match

NFILE - name of file containing entries to be cancelled

KEY - flag, set to 1 if an entry is found and cancelled, 0 otherwise

Examples

CANCEL is generally called by the verb CANCL, but it can be utilized by any module desiring to cancel an event that has already been scheduled

Statistics Collected

None.

GASP Files Used

NFILE

Permanent Attributes Accessed

In COMMON/GSPCOM/:

MX - index of the attribute of an entry in the GASP filing array that contains the index of the successor entry in the file

In COMMON/GSPFIL/:

MFE(I) - index in GASP filing array of the first entry in file I

MLC(I) - index in GASP filing array of the next entry to be removed from file number I

In COMMON/ZSERV/:

L15MSK - masking constant containing all zero bits except for fifteen one bits on the righthand end

In COMMON/ZSTACK/:

IZREF - third element of a triplet in IZHOLD, contains index of most recent entry in current stack

In blank COMMON:

SETS - GASP filing array

Program Inputs

From Calling Program.

Arguments:

DUMAT - see above

NFILE - see above

Program Outputs

To Permanent Attributes.

In COMMON/GSPFIL/

MLC(I) - index in the GASP filing array of the next entry to be removed from file I

In COMMON/ZSTACK/

IZREF - third element of a triplet in IZHOLD, contains index of most recent entry in current stack

To Calling Program.

Argument:

KEY - set to 1 if an entry is cancelled, 0 otherwise

Programs Called

Verbs. None.

Other. SET, ZRELST.

Input/Output Files Used

NPRNT - FORTRAN logical file number for printed reports.

CPLAEN

General Description

CPLAEN is used in conjunction with ZREMEN in ZSTACK to provide a FIFO stack handling capability. It is identical to ZPLAEN in all aspects but one. CPLAEN places the entry at the end of a stack so the first entry to be placed in the stack is always at the top of the stack. ZREMEN will then always remove the first entry.

IZKC is value to be placed in IZHOLD(1,I)

IZK is value to be placed in IZHOLD(2,I)

IZREF is pointer to top entry in stack in which entry is to be made.

If IZREF = 0, a new stack is formed.

IZFREE is pointer to stack of available or free entries.

IZHOLD(3,I) in each stack entry contains the pointer to the next entry in that stack. The last pointer in a stack is given the value zero. (Exception, the last pointer in the free space stack is given the value -99.

Entry point SPLAEN is a special form which sorts the entries in a stack based on the age packed in word one for a stack of shipments at a CCP.

Arguments. None.

Examples

The module CPLAEN has been used for holding stacks at a Containerization and Consolidation Point (CCP) so that the oldest shipments in a stack are removed first.

Statistics Collected

Common /CORE/

DYNCOR - statistics on IZHOLD utilization

GASP Files Used

None.

Permanent Attributes Accessed

Common /CONCOM/:

CPAK2, CPAK3 - packing factors for CCP shipments while in push-down stacks, used for checking shipment age for SPLAEN

Blank Common:

IZHOLD - array of pushdown stack triplets

Common /ZSTACK/:

IZFREE - location of top of available free space in IZHOLD

Program Inputs

From Calling Program.

Common /ZSTACK/:

IZKC - value to be placed in IZHOLD(1,I)

IZK - value to be placed in IZHOLD(2,I)

IZREF - pointer to top of stack in which entry is to be made.

Program Outputs

To Permanent Attributes.

Blank Common:

IZHOLD - triplet entry to specified stack

To Calling Program.

Common /ZSTACK/:

IXREF - pointer to top of stack which contains new entry.

Programs Called

Verbs. None.

Other. CORPT, SUMRY, OTPUT, ZERRPT, STKPR, STATI.

Input/Output Files Used

NPRNT - system output file.

DELDS
R/SS/CH
12/76

DELDS (DSTYPE, COORDM, NCOORS, NCRLST)

General Description

Routine to delete PDS datasets of type DSTYPE whose COORD values match COORDM. If COORDM(I) = -99., any value will be accepted. This can be used to release all datasets of a given type for a particular node, for example.

One special case is handled. If an MDLY dataset is released, the associated rate/level delay (RLD) datasets are released.

Arguments

DSTYPE - type of PDS dataset to be released

COORDM - coordinates to match with existing PDS datasets of type DSTYPE

NCOORS - number of coordinates for this dataset type

NCRLST - number of rate/level statistics changed

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

None.

Program Inputs

From Calling Program.

Arguments as above.

From IDSLON.

Arguments -

INDEX - location of dataset in DSPOOL array

NEL - number of elements in dataset

Common /DPTRS/:

COORD - coordinates of dataset

NCOORS - number of coordinates in COORD array

Program Outputs

To Permanent Attributes. None.

To CHRLST.

Argument

JRLST - set to -1 to turn off any rate/level statistics in datasets released

To Calling Program. None.

Programs Called

Verbs. None.

Other. CHRLST, CLEAR, IDSLON, CLER1.

Input/Output Files Used

None

DTBGC(L)

General Description

Collects garbage in DTABLE (i.e., all locations flagged by DFREED = 5HFREED in DTABLE(1.). Moves remaining tables up to give all open space at end of DTABLE. INDTBL is reset to end of entries in DTABLE. NFREDT is reset to zero. DTMAP array is MAP used to reset pointers to new DTABLE entries.

DTMAP (.1) - start of freed space removed

DTMAP (.2) - no. of entries of freed space removed thus far (offset)

IDTMAP is index to latest entry in DTMAP offset table

IOFF is no. of entries to offset the remaining entries.

This routine is called by INDTB if there are insufficient entries at the end of DTABLE for the new table but there are sufficient entries that have been freed.

Arguments.

L - number of entries needed in DTABLE.

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

COMMON /TBLCOM/:

DTABLE - an array for storing pairs of values

INDTBL - index of the last pair stored in DTABLE

DFREED - "FREE", flag for released entries in DTABLE

COMMON /DTBMAP/:

DTMAP(•,1) - original location of entry in DTABLE

DTMAP(•,2) - number of locations offset (back) from original location

IDTMAP - index of latest entry in array DTMAP

IDTMAX - maximum number of entries in DTMAP

Program Inputs

From Calling Program. None.

From RSETPA. None.

From RSETRS. None.

From RSETRD. None.

Program Outputs

To Permanent Attributes.

Common /TBLCOM/:

INDTBL - new location of last entry in DTABLE

DTABLE - same table entries relocated to utilize free space between all tables

NFREDT - 0

To Calling Program. None.

To RSETPA, RSETRS, RSETRD.

Common /DTBMAP/:

DTMAP - offset map array for DTABLE

IDTMAP - number of entries in DTMAP

Programs Called

Verbs. None.

Other. RSETPA, RSETRS, RSETRD, ZERRPT.

Input/Output Files Used

6 - system logical file for printed reports.

DTPNEW (PTROLD)

General Description

This function returns a new DTABLE pointer from a DTABLE garbage collection mapping when given an old DTABLE pointer.

Arguments.

PTROLD - old pointer to a table in DTABLE.

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

Common /DTBMAP/:

DTMAP - DTABLE mapping function from garbage collection

IDTMAP - number of entries in DTMAP array

Program Inputs

From Calling Program.

Common /DTBMAP/:

DTMAP (•,1) - original location of each table that was moved in DTABLE

DTMAP (•,2) - number of locations that the table was moved up in the DTABLE array

Program Outputs

To Permanent Attributes. None.

To Calling Program.

Function value - new pointer into DTABLE

Program Called

Verbs. None.

Other. None.

Input/Output Files Used

None.

THIS PAGE LEFT INTENTIONALLY BLANK

INDTB (ARGS, L, IP)

General Description

Input entries to DTABLE. Place L entries from array ARGS into DTABLE and return location one before start of table in IP. Collect garbage if not enough space in the open block at end of DTABL. Return IP = -1 if not enough space after G. C.

Arguments.

ARGS - array containing L entries to be input into DTABLE

L - number of entries from array ARGS to be input

IP - in DTABLE, index of the last pair occupied in DTABLE

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

COMMON/TBLCOM/:

DTABLE - array for storing pairs of values

INDTBL - index of last pair occupied in DTABLE

NTABL - total number of pairs available in DTABLE

NFREDT - number of pairs freed from tables having been released

Program Inputs

From Calling Program.

Arguments:

ARGS - array of entries to create a table in DTABLE

L - number of entries in the new table

Program Outputs

To Permanent Attributes

COMMON/TBLCOM/:

DTABLE - entries of new table

INDTBL - index of the last pair occupied in DTABLE

To Calling Program

Argument

IP - set to -1 if not enough space

- set to one location before start of new table in DTABLE otherwise

Programs Called

Verbs. None.

Other. DTBGC.

Input/Output Files Used

None.

IXZNOD (NNAM)

General Description

IXZNOD is a function that returns index of name NNAM in array ANODEL of /NODES/ or zero if not present.

Arguments.

NNAM - alpha name of a node in the model description

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

Common /NODES/:

ANODEL - array of alpha node names in the order found in the model description

Program Inputs

From Calling Program.

Argument NNAM - name of node

Program Outputs

To Permanent Attributes. None.

To Calling Program.

Function value - integer number of node in ANODEL

Programs Called

Verbs. None.

Other. None.

Input/Output Files Used

None.

THIS PAGE LEFT INTENTIONALLY BLANK

MLTEL

General Description

When used as the "operation" argument in PERMDS, multiplies the value of an element in a dataset in the PDS data structure. Control is returned to the calling routine.

Arguments

None.

Example

When CHGEL is to be used to multiply by 6 the value of the 12th element in a dataset of type DTYPE with coordinates (21, 4, 1333, -2), the calling routine should contain code with the following effect:

```
      .
      .
      .
      EXTERNAL MLTEL
      DIMENSION C(4)
      .
      .
      .
      C(1) = 21
      C(2) = 4
      C(3) = 1333
      C(4) = -2
      CALL PERMDS (MLTEL, 5HDTYPE, C, 12, 6.)
      .
      .
      .
```

MLTEL is among the routines declared EXTERNAL in common module PDSEXT.

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

In PDS Data Structure.

In /DSTOR/ - element to be multiplied

Program Inputs

From Calling Program (PERMDS). None.

Program Outputs

To Permanent Attributes.

To PDS Data Structure. The specified element multiplied.

To Calling Program (PERMDS).

In /DSNOW/ -

ERNAM - "MLTEL"

SETUP - 3

IOCOD - 2

Programs Called

Verbs. None.

Other. None.

Input/Output Files Used

None.

MSTAGE

General Description

Advances stage numbers for stage sequences whose next stage time is TNOW. If TNOW equals the n-th time value in the STAGE dataset the stage number is advanced to n+1. The routine schedules the next MSTAGE event at the next time in the STHSH sequence of all stage change times. If a stage is shorter than a specified time period, STGDEL, a subnode of ZINIT specified in ATRIB(3) will be executed so that data changes related to a stage change can be read in.

Arguments. None.

Statistics Collected

None.

GASP Files Used

File 1 - simulation time file

Permanent Attributes Accessed

COMMON /ZSTAGE/:

IZSTAG - stage sequence status array

STHSH - stage time sequence array

ISTAG - current stage in STHSH

COMMON /VRBGSP/:

ATRIB - attributes of the current MSTAGE event

TNOW - current simulation time

COMMON /DSTOR/:

DSPOOL - array where PDS datasets are stored, STAGE datasets referenced directly

COMMON /GSPCOM/:

NPRNT - logical FORTRAN I/O file number for printed reports

Program Inputs

From IDSLON

Argument IXX - index in DSPOOL of first word in STAGE dataset when all datasets have been found IXX=0 if there are no datasets IXX=-1

Argument LEN - number of elements in dataset

COMMON /DPTRS/:

IXK - indexes of current coordinates

NK - sizes of present coordinate tables

COORD - coordinates of the dataset

Program Outputs

To IDSLON

Argument STAGE - dataset type name

To FILEM

Arguments

ATRIB - attributes of the next MSTAGE event

1 - number of file to be used (time file)

Program Called

Verbs. NZINIT - initialization node in model

Other. FILEM, IDSLON.

Input/Output Files Used

NPRNT - FORTRAN logical file number for printed reports.

RDNDS

General Description

Reads PDS card types VALEL and COORD to establish a set of datasets of type DSETNC (read by RDPDS) and initial values. A VALEL card is read to obtain the initial values for the dataset elements. Then a set of COORD cards is read and for each a dataset is established using DEFDSI. This routine is called by RDPDS to handle the situation where a number of datasets of the same type have the same initial value but different coordinates. Use of this capability requires only one VALEL card for the entire set.

Arguments. None.

Examples

Initialization of statistical indices is a common occurrence that can be handled as a group of cards with the same values. As an example, the following IPDS cards would set the values for the container statistics datasets:

```
IPDS    DEFNDS   CONTNR    CONSTS    6    CONSTS
IPDS    VALEL    CONSTS      6.          8.   6.   6.   3.   1.
IPDS    COORD    CONSTS     54.         20.
IPDS    COORD    CONSTS     54.         35.
IPDS    COORD    CONSTS     54.         40.
IPDS    COORD    CONSTS     54.        463.
IPDS    COORD    CONSTS     54.       1111.
IPDS    COORD    ENDCDS
```

The request for this capability is the use of DEFNDS, rather than DEFDS, to define a number of datasets.

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

In COMMON/PDSSYS/

MAXELS - maximum number of elements a PDS dataset may contain

Program Inputs

From Calling Program.

In /BUF/ -

IBFF(3) - NELSC, number of elements in current dataset type

BFF(30) - DSID, dataset ID

From NCRDR File

VALEL cards - values of elements in datasets in IPDS VALEL card format

COORD cards - coordinates of the datasets being defined in IPDS COORD card format

Program Outputs

To Permanent Attributes. None. (DEFDSI will have stored the dataset type in PDS data structure.)

To DEFDSI

In COMMON/BUF/array containing

4-8 - VDIMC(I) I=1,2,..,5, coordinates of current dataset

9-29 - VELC(I) I=1,2,...,NELSC, initial values of elements in current dataset

To PDSERR

Arguments -

RDNDSN - "RDNDS" - name of routine detecting error

IER - error number

To Calling Program. None.

Programs Called

Verbs. None.

Other. DEFDSI, PDSERR.

Input/Output Files Used

NCRDR - FORTRAN logical file for card image input.

RLDTB (L, IP)

General Description

Release L entries from DTABLE in /TBLCOM/ starting at IP + 1. Entries are flagged by DFREED= 5HFREED for possible later garbage collection by DTBGC.

Arguments.

L - number of entries to be released

IP - zero starting point in DTABLE from which entries are to be released

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

COMMON/TBLCOM/:

DTABLE - array for storing pairs of values

NTABL - total number of pairs available

NFREDT - number of pairs freed

DFREED - "FREED"

Program Inputs

From Calling Program.

Arguments - see above

Program Outputs

To Permanent Attributes.

COMMON/TBLCOM/:

DTABLE - array for storing pairs of values with specified table released

NFREDT - number of pairs freed since last garbage collection

To Calling Program. None.

Programs Called

Verbs. None.

Other. None.

Input/Output Files Used

6 - system logical file for printed reports.

RSETPA

General Description

Reset pointers to DTABLE after garbage collection has occurred in array PARAM in /VRBGSP/. This is required since tabular distributions are referenced from PARAM (•,3) by their physical location in DTABLE. Since garbage collection alters those locations, the references must be changed. This routine is called by DTBGC.

Arguments. None.

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

COMMON /DTBMAP/:

DTMAP (•,1) - original location of entry in DTABLE

DTMAP (•,2) - number of locations offset (back) from original location

IDTMAP - index of latest entry in array DTMAP

COMMON /GSPCOM/:

NPRMS - numbers of sets of parameters in array PARAM

COMMON /VRBGSP/:

PARAM (I,1) - contains parameters for the random variables read in under index I

INDPAR (I) - a code representing the form of the random variable assigned index I, code 7 is used for tabular distributions

Program Inputs

From Calling Program.

In Common /DTBMAP/:

DTMAP - mapping function from garbage collection in DTABLE, as above

Program Outputs

To Permanent Attributes.

In COMMON /VRBGSP/

PARAM - array containing parameters of random variables with PARAM (•,3) changed for TABLE distributions

To Calling Program. None.

Programs Called

Verbs. None.

Other. None.

Input/Output Files Used

NPRNT - FORTRAN logical file for printed reports.

RSETRD

General Description

Resets rate/level boxcar DTABLE pointers after DTABLE garbage collection. These pointers are stored in array RLDPAR in common /RLSYS/.

Arguments. None.

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

Common /RLSYS/:

RLDPAR - array of rate/level delay distribution parameters

NRLDPR - number of entries in RLDPAR array

Program Inputs

From Calling Program.

Common /DTBMAP/:

Contents passed to DTPNEW

From DTPNEW.

Function value - new DTABLE pointer value

Program Outputs

To Permanent Attributes.

Common /RLSYS/:

RLDPAR (*,2) - new DTABLE pointers for boxcar type distributions

To DTNPEW.

Argument - old DTABLE pointer

To Calling Program. None.

Programs Called

Verbs. None.

Other. DTPNEW.

Input/Output Files Used

None.

THIS PAGE LEFT INTENTIONALLY BLANK

RSETRS

General Description

This module is called by DTBGC to reset DTABLE pointers from datasets LDSC, EXRAT, and FCRAT after DTABLE garbage collection. The coordinates of RATATT datasets which are keyed to the location of rate/level link stacks in DTABLE are also reset according to the mapping created by DTBGC.

Arguments. None.

Statistics Collected

None.

GASP Files Used

None.

Permanent Attributes Accessed

PDS Datasets

LDSC - linked dataset coordinates in DTABLE

EXRAT - exogenous rate stack pointers to DTABLE

FCRAT - continuous flow rate stack pointers to DTABLE

RATATT - rate stack attributes

Program Inputs

From Calling Program.

Common /DTBMAP/:

Values passed to DTPNEW

From DTPNEW.

Function value - new DTABLE pointer

From IDSLON.

Argument I - location of dataset in DSPOOL array

NE - number of elements in dataset

Program Outputs

To Permanent Attributes.

New DTABLE pointers in PDS datasets LDSC, EXRAT, FCRAT, and RATATT.

To DTPNEW.

Argument - old DTABLE pointer

To IDSLON.

Argument - the name of the dataset to loop on

To Calling Program. None.

Programs Called

Verbs. None.

Other. DTPNEW, HASHI, IDSLON, IUNPAK, MSHIFT, STORPD, ZPLAEN, ZREMEN.

Input/Output Files Used

None.

END

FILMED

10-84

DTIC